UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE CIENCIAS

Temas Selectos de Minería de Textos

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

Actuario

PRESENTA:

Sergio Daniel Raya Rios

TUTORA

Dra. Amparo López Gaona

CO-TUTORA

Dra. Lizbeth Naranjo Albarrán

Ciudad Universitaria, CD. MX., 2017.

1. **Datos del Alumno**
   Raya
   Rios
   Sergio Daniel
   5539296571
   Universidad Nacional Autónoma de México
   Facultad de Ciencias
   Actuaría
   311212361

2. **Datos del Tutor**
   Dra.
   Amparo
   López
   Gaona

3. **Datos del Co-Tutor**
   Dra.
   Lizbeth
   Naranjo
   Albarrán

4. **Datos del Sinodal 1**
   M. en C.
   José Salvador
   Zamora
   Muñoz

5. **Datos del Sinodal 2**
   Dra.
   Verónica Esther
   Arriola
   Ríos

6. **Datos del Sinodal 3**
   Mat.
   Salvador
   López
   Mendoza

7. **Datos del trabajo escrito**
   Temas Selectos de Minería de Textos
   104 páginas
   2017

Índice general

Introducción

El desarrollo tecnológico ha traído consigo muchas ventajas pero también nuevos desafíos. Cosas que eran inimaginables hasta hace algunos años ahora son posibles gracias a la tecnología. Las computadoras siguen volviéndose cada vez más potentes y controlan complejos procesos en cuestión de minutos. Según información publicada por IBM cada día creamos 2.5 quintillones de datos, tanto que el 90 % de los datos en el mundo se han creado sólo en los últimos dos años [1]. Todos estos factores en conjunto han hecho que la estadística y las ciencias de la computación combinen sus herramientas con un solo fin: extraer y resumir información. Hasta hace algunas décadas unas pocas fuentes de información se encontraban disponibles, hecho que sin duda la internet ha cambiado. Ahora basta con tener una computadora con conexión a internet para tener acceso a un sinfín de información en forma de texto, imágenes, audio y video. El hecho de que la información pueda encontrarse en diferentes formas implica un reto para las nuevas generaciones de científicos que tienen que adaptarse al constante cambio del entorno con el que interactúan.

La minería de textos ha ganado especial importancia en los últimos años dada la gran cantidad de datos que se han generado en la web a través de las redes sociales, las revistas y periódicos digitales, blogs y en general cualquier aplicación que genere texto. Además de servir como una herramienta para extraer información que ayude en la toma de decisiones, la minería de textos analiza e identifica patrones y tendencias en los datos enfocándose en aquellos modelos que sean precisos y computacionalmente eficientes [2]. Muchos de estos modelos provienen tanto de áreas clásicas como la estadística y las ciencias de la computación, así como de otras más recientes que incluyen el aprendizaje automatizado y la inteligencia artificial. Dadas las características particulares de los datos de texto, en algunos casos se tendrán que hacer ciertas modificaciones a los algoritmos que se proponen originalmente dentro de estas áreas para que puedan funcionar apropiadamente bajo este nuevo contexto [3].

El objetivo de este trabajo ha sido investigar algunas de las técnicas más importantes que se proponen dentro de la minería de textos y ejemplificarlas mediante sus aplicaciones a problemas que pueden surgir en diversos contextos. No se ha pretendido ofrecer un tratamiento profundo de estas técnicas, sino más bien abordar sus aspectos teóricos y sus propiedades más importantes. Para ello este trabajo comprende la teoría requerida para la creación de un modelo de minería de textos y algunas aplicaciones que se han propuesto a la luz de esta teoría. Dentro de la teoría se han contemplado dos enfoques principales: aprendizaje supervisado y no supervisado (esta cuestión se detalla con mayor precisión en la sección 1.5) y ha sido desarrollada en los primeros 4 capítulos. El primer capítulo ofrece un panorama amplio sobre los objetivos de la minería de textos, sus aplicaciones y sus avances más recientes. El segundo capítulo define la metodología de captura y preprocesamiento de texto usuales en minería de textos. Algunas técnicas fundamentales de análisis semántico y modelación de tópicos latentes también han sido incluidas en este capítulo y constituyen la teoría básica referente al aprendizaje no supervisado en minería de textos. El tercer capítulo consta de una recopilación de las técnicas básicas más usadas en clasificación o categorización de documentos de texto y representa el enfoque supervisado comúnmente adoptado en esta área. La teoría concluye con un breve capítulo dedicado a la evaluación del rendimiento de los algoritmos de aprendizaje. Las aplicaciones han sido organizadas en los últimos dos capítulos con el fin de ilustrar los enfoques de aprendizaje abordados anteriormente.

Las tendencias más recientes de investigación se interesan por la predicción en grandes cantidades de datos y el análisis de tendencias en las redes sociales, páginas especializadas en ventas y foros de discusión. Un método que se ha popularizado rápidamente en el área de la minería de texto consiste en la clasificación automática de documentos con connotaciones positivas o negativas. Inicialmente estos modelos eran conocidos

como análisis de sentimientos, aunque hoy en día este concepto se ha expandido y popularizado hasta convertirse en un conjunto amplio de herramientas en constante crecimiento [4]. Una aplicación referente al análisis de sentimientos se presenta en este trabajo y tiene como objetivo analizar un conjunto de opiniones recolectadas de Amazon sobre diversos productos e identificar patrones útiles en ellas que puedan ser aprovechadas en un contexto de negocio. Otro problema clásico en la minería de textos es la detección de SPAM. Este problema ha sido abordado desde muy diversas perspectivas y en este trabajo se ofrece un tratamiento estadístico como parte de las aplicaciones para resolver el problema de la detección de SPAM en teléfonos móviles.

Capítulo 1

Marco teórico de la minería de textos

La minería de textos ha ganado especial importancia por la producción masiva de datos en forma de texto presentada en los últimos años [2]. La llegada de las computadoras provocó que muchas de las actividades que las personas realizan de manera cotidiana se hayan trasladado a medios digitales. Desde la compra de un producto hasta la lectura de un libro pueden realizarse ahora sencillamente a través de una computadora. Esto implica que una gran cantidad de documentos se encuentren ahora disponibles y listos para ser analizados [5]. Dado este panorama, surgió la necesidad de crear una nueva área dedicada a analizar patrones y hacer inferencias o predicciones en este tipo especial de datos conocida hoy en día como minería de textos [2]. Este capítulo presenta una introducción al estudio de la minería de datos y pone en claro sus alcances, aplicaciones y relaciones con otras áreas.

1.1. Alcances de la minería de datos

Antes de hablar sobre la minería de textos es necesario poner en claro algunos de sus orígenes. Al ser un área que nace de un problema tan extenso, muchos de los conceptos de la minería de textos provienen de otras áreas. En particular el enfoque básico aplicado en la minería de textos proviene de un área más general conocida como minería de datos. Es por ello que antes de introducir el marco teórico de la minería de textos será conveniente hablar primero sobre los alcances de la minería de datos.

"A finales de los años ochenta surgió una nueva disciplina llamada minería de datos, dedicada a extraer conocimiento de enormes volúmenes de datos con la ayuda de una computatdora. [...] Debido a su naturaleza interdisciplinaria la minería de datos ha recibido contribuciones de muchas áreas como bases de datos, aprendizaje automatizado, estadística, recuperación de la información, visualización de datos, computación paralela y distribuida, etc." [6].

Aunque no existe una única definición sobre el concepto de minería de datos, la mayoría de los autores coinciden en algunas características que identifican esta área. Charu C. Aggarwal [3], por ejemplo, define a la minería de datos de la siguiente manera: "La minería de datos es el estudio de la recolección, limpieza, preprocesamiento, análisis y la ganancia de ideas útiles acerca de los datos" . Este mismo autor también identifica cuatro problemas fundamentales en el estudio de la minería de textos: asociación de patrones, análisis de conglomerados, clasificación y detección de valores atípicos, así como un conjunto de datos que pueden ser cuantitativos, categóricos, textos, espaciales, temporales o de orientación gráfica. Por su lado, S. Sumathi y S.N. Sivanandam [7] definen la minería de datos de la siguiente manera: "La minería de datos se puede definir como el proceso de descubrir nuevas correlaciones significativas, patrones y tendencias mediante la exploración de grandes cantidades de datos almacenados, utilizando técnicas estadísticas, de aprendizaje automatizado, inteligencia artificial y visualización de datos".

Ftrans Conen [8] sugiere que los mecanismos y las técnicas dentro del ámbito de la minería de datos pueden describirse como una amalgama de los enfoques provistos por el aprendizaje automatizado y la estadística y que de hecho una gran parte de la comunidad científica de la minería de datos está dominada principalmente por computólogos y estadísticos. Algunas de las aplicaciones más importantes en la minería de datos se mencionan

a continuación.

- Marketing: Al conocer las características de los clientes es posible implementar técnicas estratégicas de marketing con el fin de obtener un mejor impacto en las campañas [9].
- Segmentación de mercados: Identificar los gustos y cualidades de los diferentes grupos de clientes de una compañía con el fin de ofrecer servicios que se adapten a sus necesidades [7].
- Finanzas: Crear modelos de predicción que ayuden a los inversionistas en la toma de decisiones dado el comportamiento histórico de un activo en el mercado [9].
- Créditos: Diseñar un modelo automatizado de predicción que ayude a determinar cuándo un sujeto es candidato a recibir un crédito dadas las especificaciones de una organización crediticia [9].
- Abandono: Identificar los posibles clientes de la compañía que tienen más probabilidad de dejar el servicio o mudarse a otra compañía. Este problema es clásico en contextos de telefonía móvil en el que los clientes pueden cambiar con facilidad de una compañía a otra [7].
- Astronomía: Un ejemplo claro de aplicaciones en esta área lo provee el Jet Propulsion Lab del Instituto Tecnológico de California al diseñar un sistema dedicado a la clasificación de objetos espaciales, como estrellas, usando imágenes satelitales [9].
- Detección de fraudes: Dado el creciente número de fraudes bancarios, estas instituciones han desarrollado especial interés por modelos de predicción que les permitan identificar de manera oportuna transacciones fraudulentas u operaciones sospechosas [9].
- Estrategias de venta: Este problema es comúnmente conocido en la literatura como “el problema de la canasta de mercado” y consiste en determinar la disposición estratégica de los productos en una tienda de tal manera que se optimice el número de ventas [7].

1.2. ¿Qué es la minería de textos?

La minería de textos puede ser entendida de manera sencilla como una aplicación de la minería de datos que en particular analiza documentos de texto. Sin embargo, esta definición no es del todo correcta, pues hoy en día la minería de textos no es sólo una aplicación de la minería de datos sino que su alcance se ha expandido y fusionado con otros campos del conocimiento hasta consolidarse como una de las áreas más dinámicas con una amplia y activa comunidad científica dedicada a la investigación y aplicación de estas herramientas.

Al nacer de la minería de datos, la minería de textos también comparte muchas de las características interdisciplinarias de ésta. Así, “la minería de textos usa técnicas de áreas científicas bien establecidas como minería de datos, aprendizaje automatizado, procesamiento de lenguaje natural, recuperación de la información y estadística” [6]. Sobre el objetivo de la minería de textos Grobelnik et al. argumentan lo siguiente “El objetivo de la minería de textos es explotar la información contenida en documentos de texto de varias maneras, incluyendo el tipo de análisis que es típicamente realizado en minería de datos: descubrimiento de patrones y tendencias en los datos, asociaciones entre entidades, reglas predictivas, etc. Tratar con texto libre, en contraste con datos relativamente “limpios” y bien organizados, presenta nuevos retos” [10].

La mayoría de las definiciones que intentan acotar el estudio de la minería de datos coinciden en muchos aspectos. Una buena aproximación que muestra también el panorama general en la minería de textos es proporcionada por Ian H. Witten: “La minería de textos es un nuevo campo en expansión que intenta recopilar información significativa a partir del texto de lenguaje natural. Puede ser vagamente caracterizado como el proceso de analizar el texto para extraer información que es útil para propósitos particulares. En comparación con el tipo de información almacenada en las bases de datos, el texto es no estructurado, amorfo, y difícil de tratar con algoritmos. Sin embargo, en la cultura moderna, el texto es el vehículo más común para el intercambio formal de información” [11]. Frans Coenen identifica además a la minería de textos como una aplicación actual de la tecnología que va más allá de la simple “minería tabular” junto a otras aplicaciones como la minería de

imágenes y de gráficas [8].

1.3. Metodología de desarrollo de una aplicación de minería de textos

Al igual que en otras aplicaciones de la minería de datos, la minería de textos requiere un tratamiento especial de los datos que involucra aspectos de captura y recolección hasta entrenamiento y validación de los modelos. Aunque no existe una única metodología que funcione para resolver todos los problemas de minería de textos (ya que cada uno de ellos tendrá sus propias particularidades) es posible describir a grandes rasgos las características más importantes que por lo general deberá contener (implícita o explícitamente) un proceso de modelación en minería de textos (véanse por ejemplo [5] y [2]). A continuación se describen dichas características (véanse los capítulos subsecuentes para un tratamiento más extenso de cada una de ellas).

1) **Definición del problema:** Idealmente se partirá de una pregunta o problemática que se desee resolver y se deberá determinar si ésta puede ser resuelta o no usando las técnicas de modelación de minería de textos y los recursos disponibles. En muchos casos esta pregunta surge una vez que ya se han capturado los datos y se quiere obtener información de ellos. Tal caso se puede presentar con frecuencia en la práctica y se deberá determinar cuándo sería razonable volver a capturar los datos, trabajar con los datos que ya se tienen o bien reformular la pregunta. También será importante definir las herramientas que podrían ser usadas para modelar el problema y el enfoque necesario para abordarlo.

2) **Captura y recolección de los datos:** Los datos deberán ser capturados de tal manera que sean útiles para responder a la pregunta que se ha planteado. Para ello será necesario considerar primero los supuestos teóricos que requieren los modelos o el enfoque bajo el cual se pretenda resolver el problema. En muchos casos los modelos hacen supuestos de independencia y aleatoreidad en los datos y por ello será importante tomar esto en cuenta al capturarlos. La mayoría de los modelos no suelen hacer supuestos distribucionales en los datos, aunque de hacerlos se deberá validar primero la veracidad de este requisito. En muchas de las aplicaciones los datos ya se encuentran disponibles de manera electrónica, por lo general en páginas web como redes sociales y foros de discusión. En tales casos técnicas de recuperación de información como *scraping web* (véase sección 6.1.1) pueden ser aprovechadas para extraer los datos de manera automatizada.

3) **Análisis y preprocesamiento de textos:** Por lo general los datos de texto que se han capturado se encontrarán en forma de texto plano o bien en alguna otra forma no estructurada. Para que los algoritmos de aprendizaje puedan operar en ellos, es necesario transformar primero estos datos a una representación matricial (aunque algunos modelos podrían requerir otro tipo de transformaciones). Una vez que se ha obtenido esta nueva representación, los datos estarán listos para ser analizados y algunas técnicas exploratorias iniciales pueden ser aplicadas. Esta etapa suele ser una de las más laboriosas en las aplicaciones de minería de textos dado que una seríe de nuevos factores deberán ser considerados al cambiar a la nueva representación de los documentos. Estos factores dependerán de las particularidades de cada problema y deberán ser evaluados por un conjunto de expertos.

4) **Estimación y validación del modelo:** Una vez que los datos han sido cambiados a una representación adecuada para operar en ellos, se procede a ajustar los modelos propuestos. Cada uno de los modelos propuestos tendrá una serie de parámetros libres que deberán ser estimados al minimizar algún criterio de error. En algunos casos el número de parámetros podría ser tan grande que se deberá evaluar primero la viabilidad del modelo en relación a los recursos computacionales disponibles. Por lo general algunas técnicas de remuestreo deberán ser utilizadas para validar los parámetros del modelo en un conjunto restringido de ellos y la búsqueda por encontrar los mejores parámetros podría ser exhaustiva. Otros modelos no requieren de una búsqueda tan minuciosa, pues estiman sus parámetros de manera implícita, pero a pesar de ello podrían ser necesarios métodos iterativos de aproximación numérica en el proceso.

5) **Comparación y selección modelos:** Una vez que los parámetros del modelo han sido estimados de alguna manera, se procede a evaluar el rendimiento global de cada algoritmo respecto a algún criterio óptimo. Por lo general se parte de una serie de modelos que trabajan bajo diferentes principios y se desea

determinar cuál de ellos puede ser el más adecuado para resolver el problema o responder a la pregunta de interés. Al final, el algoritmo seleccionado deberá ser aquel que proporcione el mejor rendimiento global de acuerdo a alguna métrica que mida su capacidad predictiva o alguna otra cualidad que se desee que posea el algoritmo.

6) **Implementación del modelo:** Una vez que se ha determinado cuál es el modelo adecuado, se procede a implementarlo en datos reales. Si se trata de un algoritmo de predicción, se esperaría que el algoritmo cometa cierto error de acuerdo a la métrica que se ha optimizado. En cualquier caso es importante tener en cuenta la naturaleza de los datos. Por ejemplo, es importante considerar si los datos pueden cambiar en el tiempo. En tal caso es posible que nuestro algoritmo deje de ser útil en algún momento y deberá ser importante detectar cuando esto ocurra. Si se ha detectado que la naturaleza de los datos ha cambiado se deberá evaluar el impacto de estos cambios en el desempeño del algoritmo y considerar la posibilidad de reiniciar o replantear el proceso de modelación de acuerdo a los recursos disponibles.

1.4. Aplicaciones de la minería de texto

La mayoría de los autores identifican al menos cuatro grandes áreas prácticas en la minería de textos. G. Miner et al. identifica hasta siete áreas dentro de la minería de textos [12]. Algunas de las más importantes se enuncian a continuación.

- **Consulta y recuperación de información:** Consiste en un sistema automatizado de búsqueda de documentos que recibe palabras clave y produce como respuesta a la consulta un conjunto de documentos asociados a las palabras recibidas [5]. El ejemplo más sencillo de un sistema de este tipo puede ser un buscador web que muestra los resultados más relacionados a una búsqueda.

- **Análisis de conglomerados para documentos de texto:** Consiste en la identificación de patrones en los datos que pueden caracterizar a los individuos agrupándolos en conglomerados. Este escenario puede ser útil cuando una compañía desea conocer las necesidades de sus clientes a partir de una base de quejas que se encuentran capturadas en forma de texto [5]. Tal puede ser el caso de Amazon o alguna otra plataforma de ventas digitales en la que los usuarios pueden depositar sus comentarios sobre ciertos productos.

- **Clasificación de documentos de texto:** La clasificación de documentos de texto es una de las tareas clásicas en esta área. Las aplicaciones en este contexto son vastas y han tenido impacto en diferentes áreas como medicina, ciencias políticas, finanzas, sistemas computacionales, etc. La aplicación más citada de este tipo es quizás la detección de SPAM o correo no deseado. Otras aplicaciones más actuales consisten en la detección automatizada de sentimientos expresados de manera escrita a través de diferentes medios como una red social y forman parte de una de las áreas de investigación más actuales conocida como *análisis de sentimientos* o *minería de opinión* [4].

- **Minería web:** Dada la gran cantidad de datos que se encuentran en la web, éstos no pueden ser almacenados de manera inmediata y deben de ser analizados en tiempo real. En un panorama más general esta tarea es conocida como "flujos de datos" [3]. Según [12] esta aplicación debe ser realizada bajo un enfoque específico en la escala e interconexión de la web.

- **Extracción de información:** Esta área busca extraer datos de una fuente no estructurada y exportarlos hacia una base estructurada [5]. Esto puede ser útil para una organización que cuenta por ejemplo con una base de documentos de texto en forma digital y desea extraer datos de ella asociados a ventas o gastos y organizarlos de manera estructurada de tal manera que pueda analizarlos después.

- **Procesamiento de lenguaje natural:** El estudio de esta área es ampliamente investigado por campos como la inteligencia artificial y tiene como objetivo el diseño de sistemas que permitan a los humanos interactuar de manera fluida con una computadora a través del lenguaje. "Recientemente, el enfoque del preprocesamiento de lenguaje natural se ha trasladado aún más al ámbito de la minería de textos al considerar enfoques estadísticos. El procesamiento de lenguaje natural es una poderosa herramienta para proporcionar variables de entrada útiles para la minería de textos, como parte de las etiquetas de voz y los límites de frase." [12].

1.5. Sobre los alcances de este trabajo

Al igual que en el caso de la minería de datos, los algoritmos de minería de textos pueden ser clasificados en dos grandes ramas conocidas como aprendizaje supervisado y aprendizaje no supervisado. La primera de ellas se refiere a aquellos algoritmos que son entrenados a partir de una muestra que identifica a cada elemento con su respectiva clase. De esta manera el algoritmo aprende a hacer predicciones con base en una muestra previamente clasificada. En el segundo caso una muestra no etiquetada es recibida por el algoritmo y su tarea es identificar patrones en los datos de tal manera que tengan sentido para el experimentador y para el contexto del problema. Ambos enfoques serán abordados en este trabajo y las aplicaciones propuestas estarán enfocadas a ejemplificar cada uno de ellos. A continuación se describe la cobertura de este trabajo sobre estos dos enfoques.

- **Aprendizaje supervisado**: Por lo general el enfoque adoptado bajo este tipo de algoritmos es tomado de áreas como el aprendizaje automatizado con consideraciones especiales que caracterizan a las bases de texto. Es decir, los algoritmos son adaptados para resolver problemas que tienen que ver con las exigencias propias de la minería de textos. La tarea principal será por lo general la clasificación o categorización de documentos de texto. Los algoritmos considerados para esta tarea son en la mayoría de los casos máquinas de soporte vectorial, redes neuronales y clasificadores Bayesianos, entre otros. Algunas de las modificaciones esenciales tienen que ver con implementaciones más rápidas y ligeras de estos algoritmos al considerar que la clasificación de texto es por lo general un problema que implica trabajar en varias dimensiones.

  El capítulo 3 ofrece una exploración teórica sobre la clasificación de textos y contempla algoritmos como máquinas de soporte y relevancia vectorial, redes neuronales, boosting y naive Bayes. El capítulo 5 está dedicado a presentar una aplicación clásica sobre la detección de SPAM. Esta aplicación ha sido elegida por su sencillez y porque sirve para ejemplificar el tratamineto canónico de un problema de clasificación de texto desde el preprocesamiento hasta la evaluación de los algoritmos. Por su parte el capítulo 4 tiene como objetivo introducir algunas de las técnicas para evaluación de modelos más usadas en este contexto.

- **Aprendizaje no supervisado**: Los métodos más importantes de aprendizaje no supervisado son sin lugar a duda el análisis de conglomerados y la modelación de tópicos latentes. El análisis de conglomerados tiene como objetivo segmentar un corpus compuesto por un conjunto de documentos en grupos que sean lo más similares en su interior y lo más disimilares entre ellos. Por lo general este tipo de técnicas están basadas en medidas de distancia entre grupos y son ampliamente aplicadas en áreas clásicas como la estadística y la minería de datos. La modelación de tópicos latentes guarda una relación estrecha con el análisis de conglomerados y puede ser vista como una manera suave de obtener conglomerados [2]. Esto se debe a que bajo este enfoque cada documento es representado como una combinación probabilística de tópicos latentes. “De esta manera cada documento es una combinación de tópicos y cada tópico puede ser considerado como un conglomerado y la pertenencia de un documento a un conglomerado será por naturaleza probabilística, esto conlleva a representaciones de pertenencia a los conglomerados más elegantes cuando se sabe que los documentos contienen tópicos distintos [2]. Otra característica importante de los tópicos latentes es que permiten reducir la dimensionalidad en la representación de los documentos y disminuir a la vez los problemas derivados de la sinonimia y polisemia al agrupar las palabras del léxico de acuerdo a su significado semántico.

  Dadas las consideraciones anteriores, los modelos de tópicos latentes serán las únicas técnicas de aprendizaje no supervisado consideradas en este trabajo dada su importancia teórica y sus amplias aplicaciones y ventajas prácticas frente a otros algoritmos. La sección 2.6 tiene como objetivo presentar un tratamiento teórico sobre modelación de tópicos latentes. Los algoritmos más populares de este tipo consisten en variaciones del modelo de asignación Dirichlet latente [13] creado por Blei et al. en 2003. El capítulo 6 tiene como objetivo ejemplificar una aplicación sobre modelación de tópicos latentes en el contexto de la minería de opinión y el análisis de sentimientos.

1.6. Investigaciones a futuro y trabajos relacionados

Los avances más recientes en minería de textos han sido producidos en gran parte por los nuevos descubrimientos en áreas relacionadas y por el surgimiento de necesidades prácticas que antes no se tenían. La creación de nuevos conceptos de aprendizaje y una variedad de enfoques que pueden ser aprovechados en minería de textos abren la puerta a una infinidad de posibles campos fructíferos de investigación.

Un nuevo concepto conocido como aprendizaje profundo [14] ha tomado protagonismo en muchas investigaciones recientes. Bajo este enfoque se espera lograr un nuevo nivel de abstracción que permita identificar los patrones más sutiles en los datos para alcanzar una mayor precisión. Uno de los algoritmos de aprendizaje más utilizados en este contexto han sido las redes neuronales profundas. Las redes neuronales profundas consisten en múltiples capas ocultas que aprenden características abstractas de los datos de entrada al crear representaciones internas [15]. En particular, las redes neuronales convolucionales han sido aplicadas con éxito en tareas de reconocimiento de imágenes [16]. Desde luego, una gran parte de estos nuevos conceptos han sido aplicados al campo de la minería de textos (véase por ejemplo [17] y [18]). Siwei Lai et al. [19] exploran la efectividad de una red neuronal recurrente convolucional para resolver el problema de la clasificación de textos. En particular Cícero Nogueira y Maíra Gatti [20] muestran una aplicación de las redes neuronales convolucionales profundas en la que realizan un análisis de sentimientos en bases de comentarios de películas y tweets.

A pesar de ser relativamente nueva, el área del análisis de sentimientos ha atraído mucha investigación reciente. Esto se ha debido principalmente al enorme interés de las organizaciones por conocer la opinión de sus clientes y monitorear el comportamiento y las tendencias del mercado en las redes sociales y medios electrónicos. Bing Liu [4] plantea una introducción formal al análisis de sentimientos y recopila muchas de las investigaciones más actuales en el campo. La mayoría de ellas se asocian a algunos de los temas que se tratan en este trabajo, tales como modelación de tópicos latentes y categorización de documentos.

Otras aplicaciones recientes en minería de textos buscan explorar otras aproximaciones en la modelación del problema. Comúnmente se asume que el orden de las palabras en los documentos es irrelevante y entonces cada documento puede ser visto como una "bolsa de palabras". Esta conveniente representación permite simplificar el problema pero se debe estar consciente de sus limitaciones. Un aspecto importante que se omite al adoptar esta representación es la información contenida en la estructura del documento. Una frase sencilla como "no me gusta la escuela pero sí me gusta jugar" puede perder fácilmente su significado si el orden y la estructura del enunciado son descartados. De esta manera, el enunciado "sí me gusta la escuela pero no me gusta jugar" tendría exactamente las mismas palabras pero significaría lo opuesto. Nuevas áreas de investigación proponen modelar el problema aprovechando el funcionamiento interno de ciertos algoritmos de aprendizaje para preservar estas características en los documentos. Ciertos algoritmos como las máquinas de soporte vectorial, el perceptrón y el análisis de componentes principales se caracterizan porque su estructura interna depende de los datos sólo a través de productos punto y ello permite introducir el uso de funciones kernel (véase sección 3.3). La idea es entonces aprovechar esta dependencia para construir un kernel que permita preservar la estructura de los documentos al considerar a cada elemento como una cadena de caracteres en vez de una bolsa de palabras. Este nuevo enfoque es conocido como "funciones kernel de cadena" y hoy en día constituye una propuesta interesante para abordar el problema desde otro punto de vista. Huma Lodhi et al. [21] proponen el método por primera vez en el contexto de la clasificación de documentos y muestran mejoras considerables al adoptar este enfoque en comparación con el enfoque clásico. Por otro lado, Alexandros Karatzoglou e Ingo Feinerer [22] ejemplifican el uso de las funciones kernel de cadena al realizar un análisis de conglomerados a través de una implementación en el software estadístico R.

1.7. Nota bibliográfica

Aunque hoy en día existe una gran variedad de libros y documentos de investigación sobre minería de textos que pueden servir como una introducción a este tema, hay algunos de ellos que vale la pena destacar. En particular en [2] se puede encontrar un panorama amplio sobre la minería de textos que aborda problemas de clasificación, análisis de conglomerados y modelación de tópicos latentes a detalle. En [3] se enmarca la

minería de textos dentro de un conjunto de otras disciplinas que se desprenden de la minería de datos y puede servir como una introducción rápida a algunos de los aspectos más importantes en la minería de textos. Una visita básica a los preliminares de la modelación en minería de textos puede ser encontrada en [5]. Los aspectos teóricos más importantes y sus aplicaciones más clásicas han sido plasmadas en [23], mientras que algunas aplicaciones más específicas en diferentes áreas de interés han sido recopiladas en [6]. En [24] se puede encontrar una variedad de estudios de caso en minería de textos usando diversas herramientas libres tales como RapidMiner, KNIME, Python y R. Algunos de los avances y modelos más modernos en esta área han sido contemplados en [25].

Capítulo 2

Análisis y pre-procesamiento de datos

2.1. Preprocesamiento de texto

El primer paso en la minería de texto consiste en la recolección de los documentos que se pretendan analizar. Este paso es fundamental para el éxito del resto de las etapas ya que las conclusiones obtenidas dependerán directamente de la información seleccionada, por ello es importante asegurar su calidad y confiabilidad. En la mayoría de los casos el conjunto de documentos objetivo resulta ser extremadamente grande y técnicas automatizadas de muestreo tendrán que ser utilizadas para recolectar la información. En algunos contextos los documentos más recientes tendrán más importancia y el esquema de muestreo deberá considerar el tiempo asociado a cada uno de ellos. En general la muestra se verá limitada a los recursos disponibles y al error que se esté dispuesto a cometer en las conclusiones obtenidas.

Una vez que los documentos han sido seleccionados es probable que algunos de ellos se encuentren en diferentes formatos, por lo que antes de procesarlos en algún software deberán ser convertidos a un mismo formato. En particular la extensión XML (Extensible Markup Language) ha sido ampliamente adoptada como formato estándar de almacenamiento de texto por su flexibilidad como lenguaje de marcado y muchos programas de procesamiento de texto están diseñados para su uso. En algunos casos los documentos pueden incluso encontrarse como imágenes digitalizadas y algún sistema de reconocimiento óptico deberá ser implementado con el debido cuidado tomando en cuenta que éste puede cometer errores de interpretación.

Cuando los documentos han sido recolectados y convertidos a un mismo formato es posible usar algún software que permita eliminar problemas de puntuación, sinonimia y palabras poco relevantes para el diseño del algoritmo que se desee implementar. Tales problemas son comprendidos en tres métodos fundamentales:

1. *Exclusión de palabras vacías:* Las palabras que no aporten información de utilidad al diseño del método son consideradas como "palabras vacías" y deberán ser removidas del documento. Comúnmente tales palabras suelen ser artículos, preposiciones o conjunciones. Los programas especializados en procesamiento de texto tienen listas precargadas de palabras vacías en diferentes idiomas que pueden ser usadas en esta tarea. Es importante considerar que la presencia de estas palabras podría añadir ruido al proceso de decisión del algoritmo y por ello es conveniente eliminarlas.

2. *Stemming o lematización:* En general una misma palabra puede aparecer varias veces en un mismo documento ya sea porque ha sido conjugada de diferentes maneras o en diferentes tiempos, o bien porque se encuentra en plural o singular. En tales casos es conveniente consolidar todas estas variaciones en una misma palabra. Por ejemplo las palabras "biblioteca" y "bibliotecario" pueden ser consolidadas a su raíz "bibliotec". Para ello existen algunos algoritmos de stemming implementados en diferentes programas que ayudan a llevar a cabo esta tarea.

3. *Remover signos de puntuación:* Después de haber aplicado los métodos anteriores los signos de puntuación como comas y puntos deben ser eliminados. En el caso de las palabras formadas por varios vocablos unidos por un guion como por ejemplo "teórico-práctico" o "físico-químico" se deberá remover el guion si esto resulta en dos palabras significativas con diferentes connotaciones, en caso contrario podrán ser

consolidadas en una misma con la ayuda de algún diccionario digital. En la mayoría de los casos es conveniente también remover valores numéricos en caso de existir.

Los documentos que han sido procesados de acuerdo a los puntos anteriores contendrán sólo aquellas palabras con un significado semántico relevante. Cada documento es visto ahora como una "bolsa de palabras" en la que el orden es irrelevante. Esta representación resulta ser muy efectiva a pesar de la pérdida de información ocasionada al omitir el orden.

2.2. Representación documento-término

Una vez que se han estandarizado los documentos según los procedimientos expuestos en la sección anterior, se procede a adoptar una representación matricial que permita el uso de diversos algoritmos de acuerdo al problema que se tenga. En esta sección se introducirá el enfoque clásico de la minería de texto en el que una colección de documentos se ve representada por una matriz de "documento-término".

El conjunto de palabras que conforman una determinada colección de documentos es conocido como léxico. Dicha colección de documentos recibe el nombre de corpus lingüístico o simplemente corpus. La representación matricial usual permite ver a cada renglón de la matriz como un documento y a cada columna como una palabra, de tal manera que la entrada w_{ij} representa la frecuencia con que la palabra j-ésima del léxico aparece en el i-ésimo documento del corpus. De esta manera las palabras del léxico que no se encuentren en el documento tendrán frecuencia cero y la dimensión en la que cada documento se encuentra representado es igual al número total de palabras en el léxico. El resultado de esta representación es una matriz de "documento-término" $X_{n\times t}$ en la que n es el número de documentos y t el número de palabras o términos. En particular se representará al i-ésimo documento como $\bar{d}_i = (w_{i1}, ..., w_{it})$, o $\bar{d} = (w_1, ..., w_t)$ cuando el índice sea irrelevante. La matriz documento-término bajo esta notación se ve de la siguiente manera:

$$X_{n\times t} = \begin{bmatrix} \bar{d}_1 \\ \vdots \\ \bar{d}_n \end{bmatrix} = \begin{bmatrix} w_{11} & \cdots & w_{1t} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nt} \end{bmatrix}$$

Así, el conjunto de documentos $D = \{\bar{d}_1, ..., \bar{d}_n\}$ y el conjunto de términos $W = \{w_1, ..., w_t\}$ determinan la representación de la matriz $X_{n\times t}$. A lo largo de este trabajo se asumirá siempre que se parte de esta representación antes de implementar cualquier algoritmo.

La matriz $X_{n\times t}$ cuenta con algunas características clave que tienen que ser consideradas para poder adaptar ciertos algoritmos a las necesidades particulares de los datos de texto. Las características más importantes son las siguientes:

1. *Alta dimensionalidad:* Dada la variedad de palabras que pueden obtenerse de una colección de documentos la dimensionalidad del problema crece rápidamente, por lo que será necesario implementar métodos de reducción de dimensión y selección de variables con el fin de acotar el costo computacional en la medida de lo posible sin comprometer el desempeño del algoritmo.

2. *Escasez:* Un corpus puede estar compuesto de un léxico de por ejemplo 200,000 palabras, y sin embargo un documento en particular puede contener sólo unos cientos. Esto implica que muchas de las entradas de la matriz $X_{n\times t}$ serán cero. Este fenómeno es conocido como escasez en alta dimensionalidad y tiene repercusiones importantes al momento de seleccionar una medida adecuada de distancia entre documentos.

3. *No negatividad:* Las frecuencias de los documentos toman valores no negativos. Esta característica junto con las anteriores implica interesantes propiedades que pueden ser aprovechadas en el diseño y adaptación de los algoritmos. En general se debe tomar en cuenta que la presencia de una palabra es estadísticamente más significativa que su ausencia [3].

2.3. Normalización y medidas de similaridad

Dadas las características particulares de los documentos de texto, es necesario aplicar ciertas transformaciones a la matriz documento-término antes de implementar algún algoritmo. Tales transformaciones consisten en dos partes:

- *Frecuencia inversa de documento:* dos documentos que contienen una palabra que aparece con poca frecuencia pueden ser considerados más similares que aquellos que coinciden en una palabra muy común. Por esta razón las palabras menos frecuentes tendrán más importancia. Una manera común de asignar distintos pesos a los términos según su frecuencia se describe a continuación. Definimos la frecuencia inversa del i-ésimo término del léxico como:

$$id_i = \log(n/n_i)$$

donde n_i es el número de documentos que contienen el i-ésimo término, y n el total de documentos en el corpus. Note que la frecuencia inversa id_i es una función decreciente de n_i, donde $id_i = \log(n)$ si $n_i = 1$ y $id_i = 0$ si $n_i = n$. La figura 2.1 muestra esta situación.

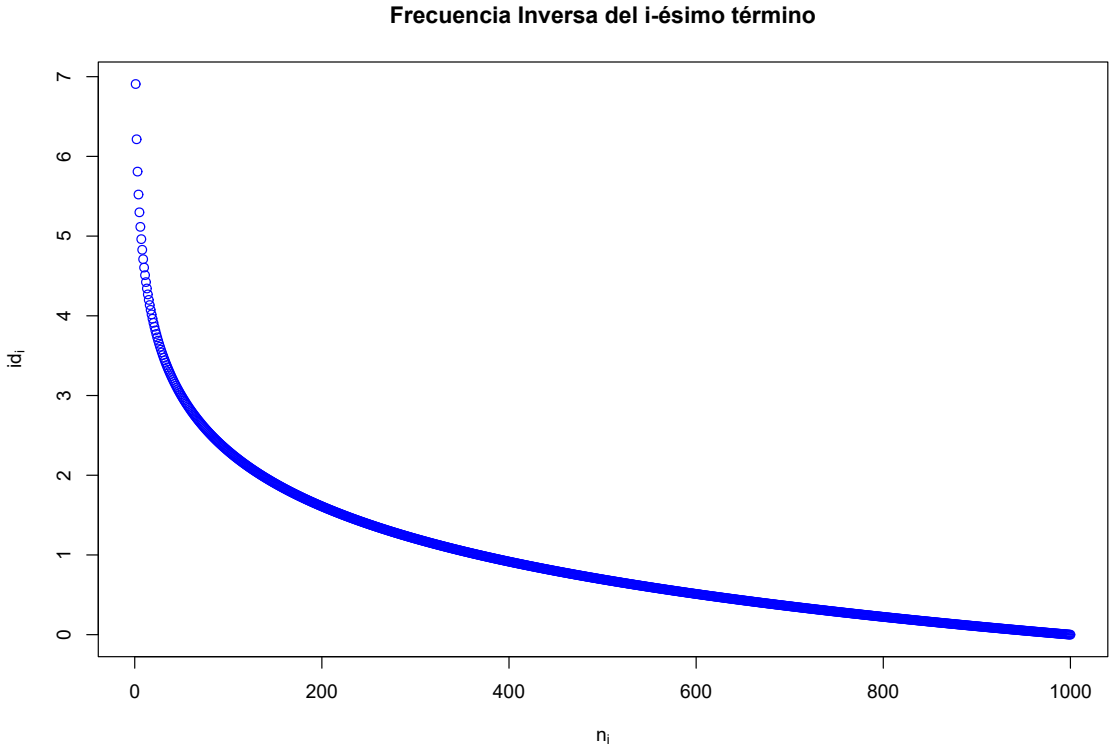


Figura 2.1: Gráfica de la función de la frecuencia inversa del i-ésimo término para n=1000.

- *Suavizamiento de la frecuencia:* En algunos casos las frecuencias de la matriz documento-término pueden ser muy diferentes entre sí, produciendo dificultades para algunos algoritmos. Una manera de homogeneizar la matriz es aplicando una función de suavizamiento a cada uno de sus términos. Las funciones más comunes en este caso son la raíz cuadrada y el logaritmo:

$$f(w_i) = \sqrt{w_i}, \tag{2.1}$$
$$f(w_i) = \log(w_i + 1). \tag{2.2}$$

En algunos casos puede no ser conveniente aplicar esta función y puede ser omitida bajo esta notación tomando $f(w_i) = w_i$.

El enfoque más común de normalización consiste en una combinación de las funciones anteriores. Definimos entonces la frecuencia normalizada $h(w_i)$ del i-ésimo término como:

$$h(w_i) = f(w_i)\, id_i$$

Este enfoque de normalización es el adoptado en los algoritmos que se presentan en este trabajo y en general en la minería de texto. Una medida de similaridad también será necesaria para calcular distancias en caso de que algún algoritmo lo requiera. Bajo este contexto el uso de la medida euclidiana no es adecuado ya que suele ser mayor en documentos largos y menor en documentos cortos, aún cuando éstos no sean similares. Se opta entonces por usar otra medida de similaridad entre documentos: sean $\bar{d_1} = (w_{11}, ..., w_{1t})$, $\bar{d_2} = (w_{21}, ..., w_{2t})$ dos documentos de un corpus que constan de un léxico con t palabras. Se define la distancia coseno entre $\bar{d_1}$ y $\bar{d_2}$ como:

$$\cos\left(\bar{d_1}, \bar{d_2}\right) = \frac{\sum_{i=1}^{t} h(w_{1i})\, h(w_{2i})}{\sqrt{\sum_{i=1}^{t} h(w_{1i})^2}\sqrt{\sum_{i=1}^{t} h(w_{2i})^2}}$$

En este contexto el uso de la distancia coseno es ideal pues mide el ángulo entre dos documentos, lo cual es insensible al tamaño del documento [3]. Aunque otras medidas de similaridad pueden ser definidas apropiadamente para este contexto, se considerará siempre la medida coseno cuando se requiera el cálculo de distancias en los algoritmos que aquí se presentan.

2.4. Nubes de palabras

Una manera rápida y sencilla de explorar los documentos de una base de texto es usando un gráfico que indique la frecuencia relativa con que se presentan ciertas palabras en cada tipo de documento. Tales gráficos son conocidos en la minería de texto como nubes de palabras. La idea es representar las frecuencias con las que aparecen las palabras más importantes en el documento haciendo variar sus atributos. En particular esto puede ser útil al momento de identificar las características que pueden guiar al algoritmo de aprendizaje a un resultado satisfactorio. Algunos de los atributos que se pueden ajustar en este tipo de representaciones son: tamaño, color, tipo de fuente y posición.

Existen muchas opciones que permiten obtener gráficos de este tipo de manera rápida y sencilla, sin embargo es importante tener en cuenta que, independientemente del software usado, primero se deberá depurar el documento con el fin de eliminar palabras redundantes o errores de puntuación. La figura 2.2 muestra una nube de palabras del famoso discurso "Tengo un sueño" pronunciado por Martin Luther King el 28 de agosto de 1963 delante del monumento a Abraham Lincoln en Washington, DC, durante una histórica manifestación de más de $200,000$ personas en pro de los derechos civiles para los negros en los Estados Unidos. Se puede apreciar cómo el discurso gira en torno a la libertad y derechos de los afroamericanos. Note además que algunos acentos y otros signos de puntuación fueron eliminados como parte del preprocesamiento.

2.5. Análisis semántico latente

Es importante tener en cuenta las limitaciones del modelo al usar la matriz documento-término para representar bases de texto. Entre las características de los documentos que se pueden perder al adoptar esta representación están aquellas asociadas a la polisemia y sinonimia. El primero de estos conceptos se refiere a las palabras que pueden tener diferentes significados según el contexto mientras que el segundo a aquellas que pueden significar lo mismo. Así, por ejemplo, dos documentos que contienen la palabra "sierra" pueden ser considerados similares cuando en realidad uno de ellos puede referirse a un sistema montañoso y otro a una herramienta. Estos problemas representan un reto para los algoritmos de clasificación de texto ya que inducen ruido en las fronteras de decisión definidas por éstos. Las técnicas que se estudian en este y los próximos capítulos pueden ayudar a reducir estos problemas y eficientar al mismo tiempo el costo computacional.

Figura 2.2: Nube de palabras del discurso pronunciado por Martin Luther King el 28 de agosto de 1963.

El análisis semántico latente (a.s.l.) es el método más fundamental de reducción de dimensión y tiene como objetivo buscar grupos semánticos de palabras que se encuentren determinados de manera implícita en los documentos. Dado que se presupone su existencia, estos grupos son considerados "latentes" y de ahí deriva el nombre de este método. El a.s.l. está basado en la descomposición en valores singulares de una matriz. Si $X_{n\times t}$ es la matriz documento-término, $D_{n\times m}$ y $T_{t\times m}$ las matrices de eigenvectores propios de XX^t y X^tX respectivamente, y $S_{m\times m}$ la matriz identidad de eigenvalores propios. Entonces la matriz X puede descomponerse como:

$$X = DST^t \tag{2.3}$$

Esta descomposición es similar a la del análisis de componentes principales con la diferencia de que en este caso es aplicada directamente a la matriz documento-término en vez de usar la matriz de varianzas y covarianzas. Considerando la ecuación (2.3), es posible reducir la dimensión de la matriz X truncando las matrices D, S y T a $k < m$ valores/vectores propios obteniendo entonces una matriz $\hat{X}$ que se aproxima a X de la siguiente manera:

$$X \approx \hat{X} = D_k S_k T_k{}^t \tag{2.4}$$

Es interesante observar que de la ecuación (2.4) se obtienen las siguientes identidades:

$$XX^t \approx \hat{X}\hat{X}^t = D_k S_k{}^2 D_k = (D_k S_k)(D_k S_k)^t \tag{2.5}$$
$$X^tX \approx \hat{X}^t\hat{X} = T_k S_k{}^2 T_k = (T_k S_k)(T_k S_k)^t \tag{2.6}$$

Las matrices de productos punto XX^t y X^tX en las ecuaciones anteriores representan las distancias entre documentos y términos respectivamente (esto ya que el producto punto entre dos vectores u y v puede ser visto como una medida de similaridad basada en el coseno del ángulo entre u y v a partir de la identidad $u \cdot v = |u||v|\cos\theta$). De estas ecuaciones es posible observar que las matrices D y T contienen en efecto los vectores propios de las matrices XX^t y X^tX. Además se puede notar que las matrices D_kS_k y T_kS_k guardan la misma estructura de distancias para los documentos y términos originales y por lo tanto pueden ser usadas para

respresentarlos en una menor dimensión. Una vez que se ha reducido la dimensión es común aplicar algún algoritmo de clasificación usando los documentos representados por la matriz $D_k S_k$ en vez de usar la matriz original $X_{n\times t}$. En particular seleccionando $k = 2$ ó $k = 3$ se obtiene una representación gráfica de los documentos que puede ser útil para identificar patrones y reconocer relaciones entre documentos y términos.

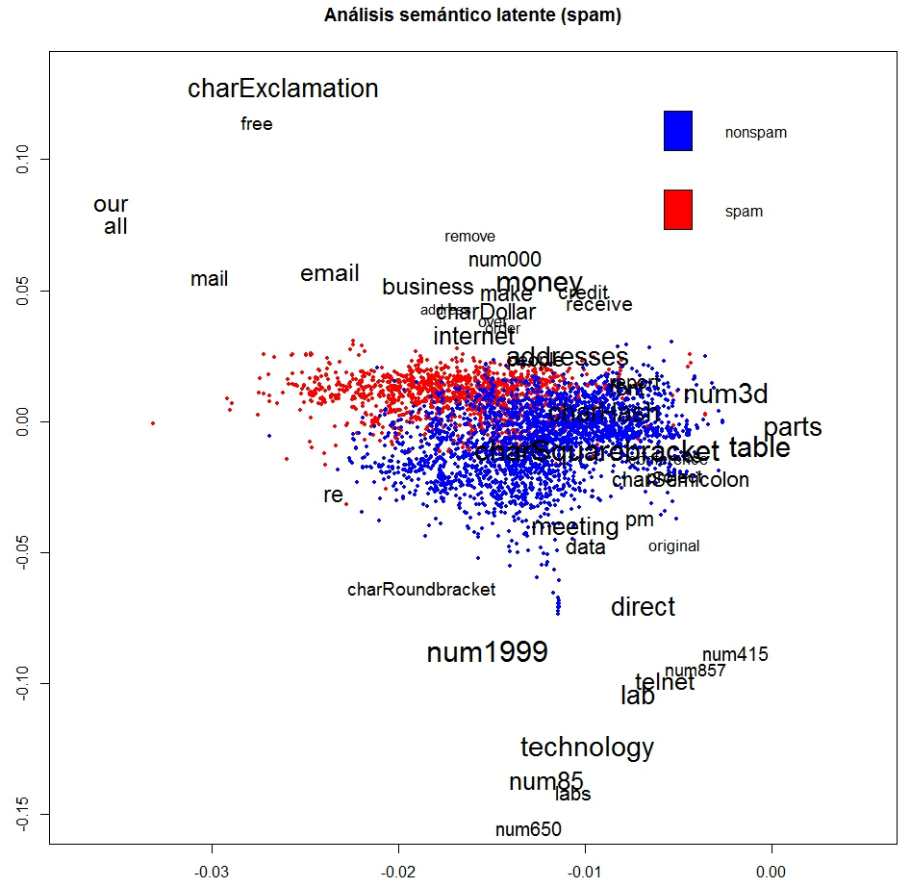


Figura 2.3: Proyección de los documentos y términos de la base de datos SPAM (*UCI repository of machine learning databases* [26]).

La figura 2.3 muestra una proyección de la base de datos SPAM en dos dimensiones obtenida al graficar los documentos y los términos a partir de los renglones de las matrices $D_k S_k$ y $T_k S_k$ respectivamente. Se puede apreciar una clara separación de los correos electrónicos clasificados como spam (puntos rojos) y no spam (puntos verdes) aunque, como en casi todo problema de clasificación, la inconveniente superposición en las fronteras dificulta la tarea para casi cualquier algoritmo. Por otro lado la figura 2.3 nos muestra también que entre las palabras más asociadas a los correos clasificados como spam se encuentran: “free”, “money”, “credit” y “business”, mientras que para los correos clasificados como no spam las palabras más asociadas son: “meeting”, “technology” y los números “1999”, “415”,“857”,“85” y “650”.

Dado que la matriz documento-término es escasa se obtendrán resultados similares usando componentes principales o aplicando un análisis semántico latente. Sin embargo, como ya se ha observado, esta última técnica permite además obtener una representación para los términos de la matriz y es aplicada frecuentemente para optimizar otros algoritmos.

2.6. Modelación de tópicos

Dado un conjunto de documentos resulta natural pensar que cada uno de ellos contiene información asociada a un tema o tópico. En particular, si pensamos en un conjunto de artículos científicos que está compuesto por una variedad de temas referentes a diversas áreas como matemáticas, física, medicina o biología, podemos preguntarnos por el tema al que pertenece cada uno de los artículos. Es claro que quizás algunos documentos puedan hablar de más de un tema a la vez, por ejemplo, un artículo sobre matemáticas puede mostrar una aplicación en áreas como biología o medicina. Si analizamos el problema con más detalle es razonable pensar además que la presencia de un tema en un documento puede condicionar la presencia de otro, por ejemplo en un artículo sobre medicina se hablará con más frecuencia de biología que de matemáticas. Dadas las observaciones anteriores se puede inferir la existencia de una estructura semántica que determina el contenido de los documentos, sin embargo tal estructura no es observada directamente y por ello se considera latente. Los métodos que se presentan en esta sección son parte de un amplio conjunto de técnicas que permiten modelar la semántica latente en los documentos y, al igual que en la sección anterior, pueden ser usadas para reducir la dimensión y remediar al mismo tiempo los problemas de sinonimia y polisemia en el léxico.

2.6.1. Análisis semántico latente probabilístico

El análisis semántico latente probabilístico (a.s.l.p.) forma parte de un grupo de métodos de reducción de dimensión para texto conocido como "topic modeling" o "modelación de tópicos". Este enfoque permite obtener resultados similares a los de la sección anterior basándose en estimaciones probabilísticas de la matriz documento-término a través del algoritmo E-M [27].

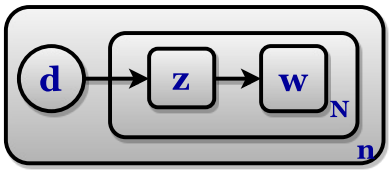


Figura 2.4: Proceso generativo del análisis semántico latente probabilístico.

El a.s.l.p. supone la existencia de k grupos o tópicos latentes $z_1, ..., z_k$. Desde un punto de vista probabilístico la entrada (i, j) de la matriz documento-término X puede modelarse como la probabilidad conjunta $P\left(\bar{d}_i, w_j\right)$ que denota la ocurrencia del documento $\bar{d}_i$ y el término w_j. Asumiendo independencia condicional de esta probabilidad conjunta dado que el tópico z_m fue observado, i.e. $P\left(\bar{d}_i, w_j \,|z_m\right) = P\left(\bar{d}_i \,|z_m\right) P\left(w_j \,|z_m\right)$, se obtiene:

$$\begin{aligned} P\left(\bar{d}_i, w_j\right) &= \sum_{m=1}^{k} P(z_m) P\left(\bar{d}_i, w_j \,|z_m\right) \\ &= \sum_{m=1}^{k} P(z_m) P\left(\bar{d}_i \,|z_m\right) P\left(w_j \,|z_m\right) \end{aligned} \tag{2.7}$$

El objetivo bajo este modelo es entonces estimar las probabilidades $P(z_m)$, $P\left(\bar{d}_i \,|z_m\right)$ y $P\left(w_j \,|z_m\right)$ para luego aproximar la matriz documento-término X a partir de la ecuación (2.7). A continuación se describe el algoritmo de Esperanza-Maximización (E.M.) que permite estimar dichas probabilidades.

- *Paso-E:* Se obtiene la estimación de la probabilidad condicional $P\left(z_m \left|\bar{d}_i, w_j\right.\right)$ en términos de las probabilidades $P(z_m)$, $P\left(\bar{d}_i \,|z_m\right)$ y $P\left(w_j \,|z_m\right)$. Usando el Teorema de Bayes, el supuesto de independencia condicional y la ecuación (2.7) se obtiene:

$$P\left(z_m \left|\bar{d}_i, w_j\right.\right) = \frac{P(z_m) P\left(\bar{d}_i \,|z_m\right) P\left(w_j \,|z_m\right)}{\sum_{m=1}^{k} P(z_m) P\left(\bar{d}_i \,|z_m\right) P\left(w_j \,|z_m\right)} \tag{2.8}$$

  El algoritmo se inicializa suponiendo una distribución uniforme para cada probabilidad, i.e. $P(z_m) = \frac{1}{k}$, $P\left(\bar{d}_i \,|z_m\right) = \frac{1}{n}$, $P\left(w_j \,|z_m\right) = \frac{1}{t}$; donde k es el número de tópicos, n el número de documentos en el corpus y t el número de términos en el léxico.

- *Paso-M:* Sea $f\left(\bar{d_i}, w_j\right)$ la frecuencia del término w_j en el documento $\bar{d_i}$, se procede a estimar cada una de las probabilidades obtenidas en el paso-E en términos de la probabilidad $P\left(z_m \left| \bar{d_i}, w_j \right.\right)$ de la siguiente manera:

$$P\left(\bar{d_i} \left| z_m \right.\right) \propto \sum_{w_j} f\left(\bar{d_i}, w_j\right) P\left(z_m \left| \bar{d_i}, w_j \right.\right) \tag{2.9}$$

$$P\left(w_j \left| z_m \right.\right) \propto \sum_{\bar{d_i}} f\left(\bar{d_i}, w_j\right) P\left(z_m \left| \bar{d_i}, w_j \right.\right) \tag{2.10}$$

$$P\left(z_m\right) \propto \sum_{\bar{d_i}} \sum_{w_j} f\left(\bar{d_i}, w_j\right) P\left(z_m \left| \bar{d_i}, w_j \right.\right) \tag{2.11}$$

Al alcanzar la convergencia, el algoritmo E-M descrito anteriormente produce como resultado tres conjuntos de probabilidades que son dispuestos en forma matricial de la siguiente manera:

$$D_{n\times k} = \left[P\left(\bar{d_i} \left| z_m \right.\right)\right]_{i=1,\ldots,n}^{m=1,\ldots,k} \tag{2.12}$$

$$S_{k\times k} = diag\left(P\left(z_m\right)\right) \tag{2.13}$$

$$T_{t\times k} = \left[P\left(w_j \left| z_m \right.\right)\right]_{j=1,\ldots,t}^{m=1,\ldots,k} \tag{2.14}$$

Observe que según la ecuación (2.7) y dado que el modelo asume que $X_{n\times t} = \left[P\left(\bar{d_i}, w_j\right)\right]_{i=1,\ldots,n}^{j=1,\ldots,t}$, se obtiene la siguiente identidad matricial:

$$X \approx \hat{X} = D_k S_k {T_k}^t$$

Se ha conservado la notación usada en la versión no probabilística del análisis semántico latente presentada en la sección anterior con el fin de evidenciar la correspondencia entre ambos enfoques. Al igual que en el a.s.l., la representación de los documentos en la nueva dimensión reducida estará dada por la matriz $D_k S_k$. La figura 2.4 ilustra el proceso generativo del análisis semántico latente probabilístico. Las cajas representan el número de veces que se repite el proceso que contienen. Los círculos contienen variables continuas y los cuadrados variables categóricas.

2.6.2. Asignación Dirichlet latente

La asignación Dirichlet latente (a.d.l.) es quizá el método más popular y más usado dentro de la modelación de tópicos, fue introducido en 2003 por David M. Blei, Andrew Y. Ng y Michael I. Jordan [13] considerando un enfoque Bayesiano que permite ganar flexibilidad y disminuir a la vez el número de parámetros. Este modelo asume que cada documento puede ser visto como una mezcla de tópicos latentes dada por una distribución inicial Dirichlet y de ahí recibe su nombre. La estructura del modelo se compone de una jerarquía de tres niveles con dos distribuciones Dirichlet y dos distribuciones discretas que por lo general son multinomiales. El proceso de construcción del método es generativo y procede de la siguiente manera:

- Para cada documento $\bar{d_i}$ con $i = 1, \ldots, n$ en el corpus $D = \left\{\bar{d_1}, \ldots, \bar{d_n}\right\}$:

  1 *Dado que el proceso es generativo, la longitud del documento puede considerarse una variable aleatoria también. Así, el primer paso consiste en elegir la longitud t_i del documento i según la distribución:*

  $$t_i \sim Poisson\left(\xi\right)$$

  2 *Elegir una distribución inicial $\theta^{\left(\bar{d_i}\right)}$ de los tópicos en el documento:*

  $$\theta^{\left(\bar{d_i}\right)} \sim Dirichlet\left(\alpha\right)$$

- *Para cada una de las palabras w_j con $j = 1, \ldots, t_i$ en el documento:*

3 *Elegir uno de los k-tópicos disponibles "z_i" con $i = 1,...,k$ para cada una de las $N^{(\bar{d}_i)}$ palabras de acuerdo a la distribución:*

$$z_i|\theta^{(\bar{d}_i)} \sim Multinoulli\left(\theta^{(\bar{d}_i)}\right)$$

4 Finalmente se elige una palabra en particular correspondiente al tópico asignado de acuerdo a la distribución:

$$w_i|z_i,\, \phi^{(z)} \sim Multinoulli\left(\phi^{(z)}\right)$$

donde $\phi^{(z)}$ es la distribución asociada a cada tópico z_i y está dada de manera inicial por:

$$\phi^{(z)} \sim Dirichlet\,(\beta)$$

La distribución multinoulli [28] es simplemente una multinomial de parámetro $n = 1$ y hace referencia a una distribución Bernoulli ya que por ejemplo si se obtiene el cuarto de cinco tópicos posibles entonces se representará este resultado por el vector dummy $(0,0,0,1,0)$. Formalmente la densidad multinoulli está definida para el j-ésimo tópico del i-ésimo documento de la siguiente manera:

$$Multinoulli\,(z_{ij}\,|\theta_i) = \prod_{l=1}^{k} \theta_{il}{}^{I\left(z_{ij}=l\right)}$$

donde

$$I\,(z_{ij} = k) = \begin{cases} 1 & \text{si el tópico } k \text{ es asignado en } z_{ij} \\ 0 & e.o.c. \end{cases}$$

Siguiendo el procedimineto anterior el algoritmo busca encontrar los tópicos latentes que maximizan la probabilidad de generar los documentos originales a partir de ellos. Suponga por ejemplo que se tiene una colección de artículos científicos y que se desea obtener tres tópicos latentes. El algoritmo genera tres colecciones semánticas de palabras que pueden tener un significado en común. Si las colecciones o tópicos generados son "matemáticas", "física" y "química", entonces los documentos pueden ser vistos como mezclas de estos temas, por ejemplo un documento puede hablar un 70 % de matemáticas, un 20 % de física y un 10 % de química. En este caso podemos representar a este documento en el espacio simplejo de dos dimensiones [1] como el vector $(0.7,0.2,0.1)$. Así, cualquier documento es visto como un vector de entradas positivas que suman uno y por lo tanto pertenece al simplejo en $k-1$ dimensiones.

Esta conveniente representación permite además tratar los problemas asociados a la polisemia y sinonimia de las palabras. Si por ejemplo se tienen las palabras "lentes" y "anteojos", éstas podrán ser agrupadas en un mismo tópico dado que su significado semántico es el mismo, reduciendo así el problema de la sinonimia provocado por estas palabras. Por el contrario, si la palabra "banco" se refiere a una entidad financiera dentro de un contexto y a un asiento en otro, el problema podrá ser reducido al considerar la palabra en ambos tópicos.

Es importante notar que al representar a cada documento como una "bolsa de palabras", no se está asumiendo independencia ya que por ejemplo el hecho de contener la palabra "matemáticas" hace más probable observar la palabra "teorema" en el mismo documento. El supuesto bajo esta representación es menos restrictivo que la independencia y formalmente se conoce como *intercambiabilidad*, este supuesto hace ideal la modelación con el enfoque Bayesiano [29], [30].

La estimación de los parámetros en el modelo se puede llevar a cabo de diferentes maneras. Originalmente se propone un método conocido como inferencia variacional [13], sin embargo una serie de algoritmos numéricos y de simulación han surgido para ofrecer estimaciones alternativas. En particular el método que se presentará a continuación fue desarrollado por Thomas L. Griffiths y Mark Steyvers [31] vía métodos de Monte

[1] el espacio simplejo de k dimensiones o espacio k-simplejo está formado por aquellos vectores de k entradas no negativas que suman uno.

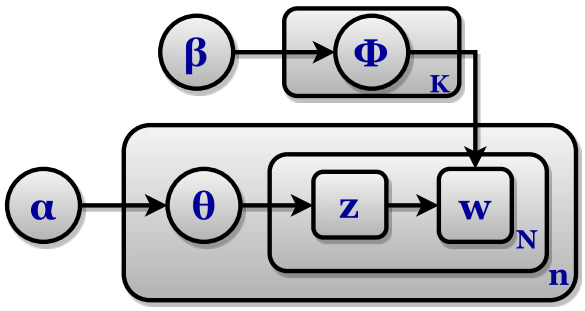


Figura 2.5: Jerarquías del modelo de asignación Dirichlet latente.

Carlo (MCMC). Se propone implementar un algoritmo "Gibbs sampler-colapsado" donde los parámetros Φ y θ son excluidos de la distribución objetivo integrando sobre ellos, esto permite disminuir la varianza al mismo tiempo que reduce el costo computacional al muestrear en un espacio parametral más reducido [28]. El objetivo es entonces estimar los tópicos latentes $z = (z_1, ..., z_k)$ dadas las palabras observadas $w = (w_1, ..., w_t)$ muestreando de la distribución:

$$P(z|w) = \frac{P(z,w)}{P(w)} = \frac{P(z,w)}{\sum_z P(z,w)} = \frac{P(w|z)P(z)}{\sum_z P(w|z)P(z)} \tag{2.15}$$

donde las densidades $P(w|z)$ y $P(z)$ son obtenidas colapsando los parámetros Φ y θ como se muestra a continuación.

La notación introducida para efecto de estos cálculos puede resultar complicada por el número de índices involucrados pero resulta intuitiva si se tiene en cuenta que el proceso generativo es primero a nivel documento y después a nivel término. La densidad conjunta de los tópicos asignados a las palabras en todos los documentos se calcula de la siguiente manera.

Sean $n_k^{(i)}$ el número de palabras en el i-ésimo documento que son asignadas al k-ésimo tópico y t_i el número total de palabras en el documento, entonces:

$$n_k^{(i)} = \sum_{j=1}^{t_i} I(z_{ij} = k) \tag{2.16}$$

$$t_i = \sum_{l=1}^{k} n_l^{(i)} \tag{2.17}$$

donde como antes

$$I(z_{ij} = k) = \begin{cases} 1 & \text{Si el tópico } k \text{ es asignado en } z_{ij} \\ 0 & e.o.c. \end{cases}$$

Así, de las propiedades de probabilidad condicional, la definición de la densidad multinoulli, las ecuaciones (2.16) y (2.17) y marginalizando respecto de θ se obtiene:

$$
\begin{aligned}
P(z) &= \prod_{i=1}^{n} \int P(z_i, \theta_i)\, d\theta_i = \prod_{i=1}^{n} \int P(z_i|\theta_i)\, P(\theta_i)\, d\theta_i \\
&= \prod_{i=1}^{n} \int \left(\prod_{j=1}^{t_i} Multinoulli(z_{ij}|\theta_i) \right) \cdot Dir(\theta_i)\, d\theta_i \\
&= \prod_{i=1}^{n} \int \left(\prod_{j=1}^{t_i} \left(\prod_{l=1}^{k} {\theta_{il}}^{I(z_{ij}=l)} \right) \right) \cdot \left(\frac{\Gamma(k\alpha)}{\Gamma(\alpha)^k} \prod_{l=1}^{k} {\theta_{il}}^{\alpha-1} \right) d\theta_i \\
&= \left(\frac{\Gamma(k\alpha)}{\Gamma(\alpha)^k} \right)^n \prod_{i=1}^{n} \int \left(\prod_{j=1}^{t_i} \left(\prod_{l=1}^{k} {\theta_{il}}^{I(z_{ij}=l)} \right) \right) \left(\prod_{l=1}^{k} {\theta_{il}}^{\alpha-1} \right) d\theta_i \\
&= \left(\frac{\Gamma(k\alpha)}{\Gamma(\alpha)^k} \right)^n \prod_{i=1}^{n} \int \left(\prod_{l=1}^{k} {\theta_{il}}^{\sum_{j=1}^{t_i} I(z_{ij}=l)} \right) \left(\prod_{l=1}^{k} {\theta_{il}}^{\alpha-1} \right) d\theta_i \\
&= \left(\frac{\Gamma(k\alpha)}{\Gamma(\alpha)^k} \right)^n \prod_{i=1}^{n} \int \left(\prod_{l=1}^{k} {\theta_{il}}^{\sum_{j=1}^{t_i} I(z_{ij}=l)+\alpha-1} \right) d\theta_i \\
&= \left(\frac{\Gamma(k\alpha)}{\Gamma(\alpha)^k} \right)^n \prod_{i=1}^{n} \int \left(\prod_{l=1}^{k} {\theta_{il}}^{n_l^{(i)}+\alpha-1} \right) d\theta_i \\
&= \left(\frac{\Gamma(k\alpha)}{\Gamma(\alpha)^k} \right)^n \prod_{i=1}^{n} \frac{\prod_{l=1}^{k} \Gamma\left(n_l^{(i)}+\alpha\right)}{\Gamma\left(\sum_{l=1}^{k} \left(n_l^{(i)}+\alpha\right)\right)} \\
&= \left(\frac{\Gamma(k\alpha)}{\Gamma(\alpha)^k} \right)^n \prod_{i=1}^{n} \frac{\prod_{l=1}^{k} \Gamma\left(n_l^{(i)}+\alpha\right)}{\Gamma\left(\sum_{l=1}^{k} n_l^{(i)}+k\alpha\right)} \\
&= \left(\frac{\Gamma(k\alpha)}{\Gamma(\alpha)^k} \right)^n \prod_{i=1}^{n} \frac{\prod_{l=1}^{k} \Gamma\left(n_l^{(i)}+\alpha\right)}{\Gamma(t_i+k\alpha)}
\end{aligned}
\tag{2.18}
$$

$$
\therefore P(z) = \left(\frac{\Gamma(k\alpha)}{\Gamma(\alpha)^k} \right)^n \prod_{i=1}^{n} \frac{\prod_{l=1}^{k} \Gamma\left(n_l^{(i)}+\alpha\right)}{\Gamma(t_i+k\alpha)} \tag{2.19}
$$

Análogamente se puede probar que si $n_k^{(w)}$ es el número de veces que la palabra w es asignada al tópico k y si $n_k^{(\cdot)} = \sum_{w=1}^{t} n_k^{(w)}$ es el número total de palabras asignadas al tópico k, entonces:

$$
P(w|z) = \left(\frac{\Gamma(t\beta)}{\Gamma(\beta)^t} \right)^k \prod_{l=1}^{k} \frac{\prod_{w=1}^{t} \Gamma\left(n_l^{(w)}+\beta\right)}{\Gamma\left(n_l^{(\cdot)}+t\beta\right)} \tag{2.20}
$$

Finalmente, para implementar el algoritmo de Gibbs, las condicionales completas son obtenidas a partir de las densidades (2.19) y (2.20). Así, considerando que $\Gamma(x+1)/\Gamma(x) = x$ se tiene:

$$
p\left(z_{i,j} = k \,\middle|\, z_{-i,-j}, w\right) \propto \frac{n_k^{(w_{-j})}+\beta}{n_k^{(\cdot,-j)}+t\beta} \cdot \frac{n_k^{(i,-j)}+\alpha}{t_{i,-j}+k\alpha} \tag{2.21}
$$

Donde $n_k^{(w_{-j})}$, $n_k^{(\cdot,-j)}$, $n_k^{(i,-j)}$ y $t_{i,-j}$ tienen los significados definidos anteriormente pero sin considerar la palabra $w_{i,j}$ en cada conteo. Note que de esta manera la densidad (2.21) tiene una interpretación muy intuitiva: el primer

factor representa la frecuencia con la que la palabra $w_{i,j}$ es generada por el tópico k y el segundo factor indica la frecuencia con que el tópico k es observado en el documento i. En otras palabras el algoritmo actualiza las configuraciones de los tópicos en cada iteración preguntándose: ¿con qué probabilidad se genera esta palabra en el tópico? y ¿con qué probabilidad se observa este tópico en el documento?

El algoritmo se inicializa asignando tópicos $z_{i,j} \in \{1,...,k\}$ (como estado inicial de la cadena de Markov) de manera aleatoria a cada palabra en cada documento y luego se muestrea de la distribución (2.21) hasta alcanzar la convergencia. El algoritmo puede operar de manera eficiente con poca memoria ya que sólo necesita actualizar los conteos de las palabras en los tópicos y en los documentos en cada iteración. Una vez que se ha alcanzado la convergencia el último estado de la cadena es capturado y las muestras subsecuentes son generadas con un adecuado retraso para asegurar que la autocorrelación es baja [31]. Una vez obtenida una muestra de la densidad (2.21) es posible derivar estimaciones de los parámetros posteriores de las distribuciones Dirichlet de las palabras y los documentos para cada tópico $l = 1,...,k$ [31], según las ecuaciones:

$$\hat{\Phi}_l^{(w)} = \frac{n_l^{(w)} + \beta}{n_l^{(.)} + t\beta} \tag{2.22}$$

$$\hat{\theta}_l^{(d)} = \frac{n_l^{(i)} + \alpha}{t_i + k\alpha} \tag{2.23}$$

En particular el vector $\hat{\theta}^{(d)} = \left(\hat{\theta}_1^{(d)},...,\hat{\theta}_k^{(d)}\right)$ puede ser visto como una respresentación en k dimensiones del documento d en el espacio simplejo de $k-1$ y representa las proporciones que definen el documento como una mezcla de los diferentes tópicos.

2.6.3. Modelo tópico correlacionado

Es importante tomar en cuenta que la presencia de un tópico en un documento puede condicionar de manera importante la presencia de otro, por ejemplo si un documento habla sobre física sería natural pensar que en particular hable sobre matemáticas también. Una de las principales desventajas de la asignación Dirichlet latente es que no permite considerar la estructura de correlación entre tópicos. El modelo tópico correlacionado fue introducido por David M. Blei y John D. Lafferty [32] como una respuesta a este problema.

El modelo supone una distribución normal logística en vez de la distribución Dirichlet con el fin de mantener la representación de los documentos en el simplejo considerando a la vez la manera en que correlacionan los tópicos entre ellos. En este caso una implementación del algoritmo de Gibbs es infactible dada la complejidad de la distribución normal logística y por ello se opta por métodos de inferencia variacional [32]. Una implementación del algoritmo Metropolis-Hastings [33] es posible pero computacionalmente inadecuada dada la dimensionalidad del problema.

Para obtener el vector de proporciones inicales de cada tópico en el documento w se genera primero un vector η de la distribución normal k-variada:

$$\eta \sim N(\mu_k, \Sigma_{k\times k})$$

Se aplica al vector $\eta = (\eta_1,...,\eta_k)$ la transformación $f(\eta_l)$ que mapea al simplejo para $l = 1,...,k$:

$$f(\eta_l) = \frac{\exp\{\eta_l\}}{\sum_{i=1}^{k} \exp\{\eta_i\}} \tag{2.24}$$

Así, el vector de proporciones para los tópicos en cada documento está definido por:

$$\theta = (f(\eta_1),...,f(\eta_k)) \tag{2.25}$$

Ésta es la única modificación al modelo a.d.l., el resto del proceso generativo se realiza de la misma manera.

2.6.4. Modelo tópico dinámico

En algunos casos es importante considerar que los documentos evolucionan a través del tiempo, por ejemplo las palabras usadas en cada tópico de un conjunto de artículos científicos pueden cambiar según la evolución del pensamiento y los nuevos descubrimientos tecnológicos. Las redes sociales son un ejemplo en el que este tipo de evolución puede ser muy rápida y cambiar según las tendencias más novedosas. El modelo tópico dinámico [34] permite considerar este cambio en el tiempo al introducir un ruido Gaussiano al proceso de evolución de la distribución de cada tópico a través del tiempo.

Asumiendo que la distribución de las palabras en el k-ésimo tópico evoluciona en el conjunto de variables aleatorias indexadas en el tiempo $\left\{\vec{\pi}_{1,k},...,\vec{\pi}_{T,k}\right\}$ de acuerdo a un proceso:

$$\vec{\pi}_{t,k}\left|\vec{\pi}_{t-1,k}\right. \sim N\left(\vec{\pi}_{t-1,k},\sigma^2 I\right)$$

Observe que la notación anterior significa que condicionada al valor que ha tomado la variable $\vec{\pi}_{t-1,k}$, la variable $\vec{\pi}_{t,k}$ sigue una distribución $N\left(\vec{\pi}_{t-1,k},\sigma^2 I\right)$. El algoritmo generativo procede de la misma manera que en el modelo a.d.l. excepto que en cada tiempo se considera la distribución de la variable aleatoria $\vec{\pi}_{1,k}$ actualizada respecto de $\vec{\pi}_{t-1,k}$ en vez de la distribución Dirichlet ϕ usada para modelar la distribución de las palabras en cada tópico. Al igual que en el modelo tópico correlacionado se deberá mapear esta distribución al espacio simplejo usando la función (2.24) en cada tiempo. Así, se tendrá una colección secuencial de modelos tópicos indexados por los tiempos $t = 1,...,T$.

Los resultados obtenidos por el modelo pueden ser presentados en una gráfica que muestre la evolución de la distribución de las palabras de cada tópico a través del tiempo. Ashok N. Srivastava y Mehran Sahami [25] proponen usar las distribuciones posteriores $\vec{\widehat{\pi}}^{w}_{t,k}$ para cada palabra w y cada tópico k al tiempo t para calcular una medida "$term-score$" que refleje la importancia evolutiva de los términos a través del tiempo definida como:

$$term-score_{k,w}(t) = \vec{\widehat{\pi}}^{w}_{t,k} \cdot \log\left(\frac{\vec{\widehat{\pi}}^{w}_{t,k}}{\left(\sum_{j=1}^{T}\vec{\widehat{\pi}}^{w}_{j,k}\right)^{\frac{1}{k}}}\right) \tag{2.26}$$

Calculando (2.26) para cada palabra se obtiene una gráfica que muestra el comportamiento evolutivo por importancia de cada término y éste puede ser comparado con eventos históricos que pudieron haber influido en dicha evolución. Otras adaptaciones del modelo a.d.l. pueden ser implementadas y generalmente el método variacional [34] dará los mejores resultados.

2.6.5. Selección de parámetros

Una parte importante en la modelación de tópicos es la selección de parámetros en el modelo. El interés se enfoca en seleccionar los parámetros α y β correspondientes a las distribuciones Dirichlet a priori y el número de tópicos latentes k. Generalmente los paquetes estadísticos determinan por default los valores para los parámetros α y β. Griffiths y Steyvers [31] proponen por ejemplo un valor de $50/k$ para α y 0.1 para β [31]. Originalmente David M. Blei et. al. [13] maximizan la cota inferior de la log-verosimilitud del modelo respecto de los parámetros α y β usando inferencia variacional (véase [13]). Aunque una búsqueda en una retícula de parámetros (o "*grid*") puede dar uno de los mejores resultados, puede consumir mucho tiempo [2]. Generalmente el interés se centra en determinar el número de tópicos ya que de esto dependerá en gran medida la calidad del modelo. La búsqueda del número de tópicos óptimo es exhaustiva y requiere de entrenar un número considerable de modelos para calcular después una medida de bondad. La mayoría de las medidas están basadas en nociones de disimilaridad entre tópicos y aproximaciones a la verosimilitud del modelo. A continuación se describen algunas de las medidas más usadas en modelación de tópicos latentes. Las primeras cuatro han sido definidas originalmente para el modelo de asignación Dirichlet latente mientras que las últimas

dos son usadas universalmente para evaluar la calidad de modelos de procesamineto de lenguaje natural y modelación de tópicos.

2.6.5.1. Método de la media armónica (Griffiths y Steyvers)

El método de la media armónica fue introducido por Griffiths y Steyvers [31] en 2004. Debido a su simplicidad es ampliamente usado para determinar el número óptimo de tópicos latentes en el modelo. Aunque se ha documentado su inestabilidad en algunos trabajos, es una buena aproximación dado su bajo costo computacional. El método está basado en los resultados obtenidos por Kass y Raftery en 1995 [35]. Dado un problema de inferencia Bayesiana con una distribución a priori $\pi(\theta)$ y un modelo $P(D|\theta)$, es posible aproximar $P(D) = \int P(D|\theta)\pi(\theta)d\theta$ con la media armónica:

$$\widehat{P(D)} = \frac{1}{\frac{1}{m}\sum_{i=1}^{m} P\left(D|\theta^{(i)}\right)^{-1}} \tag{2.27}$$

El estimador (2.27) converge casi seguramente a $P(D)$ cuando $m \to \infty$ [35]. En este caso el objetivo es maximizar la verosimilitud $P(w|k) = \int P(w|z,k)\pi(z)dz$, por lo que de (2.27) se obtiene la siguiente estimación:

$$\widehat{P(w|k)} = \frac{1}{\frac{1}{m}\sum_{i=1}^{m} P\left(w|z^{(i)},k\right)^{-1}} \tag{2.28}$$

Los valores de la densidad $P\left(w|z^{(i)},k\right)$ son obtenidos de la ecuación (2.20) al muestrear de la densidad objetivo con el algoritmo de Gibbs **??**. En este caso se seleccionará el valor de k que maximice la verosimilitud.

2.6.5.2. Distancia promedio (Juan Cao)

Este método fue propuesto por Juan Cao et al. (2009) [36] y está basado en la noción de distancia entre tópicos latentes. Bajo este enfoque se piensa que "en una buena estructura tópica de un modelo de asignación Dirichlet latente, cada tópico es un conglomerado (*cluster*) semántico comprensible, significativo y compacto, y es ajeno a cualquier otro" [36], esto significa que se tendría que generar un nuevo tópico siempre que los tópicos actuales no cumplan con dichas características. Se propone usar la distancia coseno definida en la sección 2.3 para cuantificar la ción entre tópicos:

$$\cos(z_i, z_j) = \frac{\sum_{l=1}^{t} z_{il} \cdot z_{jl}}{\sqrt{\sum_{l=1}^{t} z_{il}^2}\sqrt{\sum_{l=1}^{t} z_{jl}^2}} \tag{2.29}$$

Entre más pequeño sea el valor de (2.29) se tendrá mayor independencia entre tópicos. Finalmente se toma el promedio de las distancias entre cada par de tópicos como medida de la estabilidad en la estructura del modelo:

$$dis.prom.(estructura) = \frac{\sum\sum_{i<j}\cos(z_i, z_j)}{\left(\frac{k(k-1)}{2}\right)} \tag{2.30}$$

El número óptimo de tópicos será aquel que minimize (2.30).

2.6.5.3. Medida basada en la divergencia simétrica de Kullback-Leibler (R. Arun)

R. Arun et al. [37] proponen en 2010 una medida obtenida como resultado de sus investigaciones y experimentos empíricos basada en la divergencia simétrica de Kullback-Leibler. El método asume implícitamente que se cuenta con un léxico suficientemente grande y ha sido demostrada su efectividad en problemas de reducción de dimensión para imágenes y en texto [37].

Sean M_1, M_2 las matrices que contienen las distribuciones Dirichlet posteriores para las palabras en los tópicos y los tópicos en los documentos, respectivamente. Se propone la siguiente medida para encontrar el número óptimo de tópicos:

$$m(M_1, M_2) = KL(C_{M_1}|C_{M_2}) + KL(C_{M_2}|C_{M_1}) \tag{2.31}$$

donde C_{M_1} es el vector que contiene los valores propios de M_1, C_{M_2} es la distribución global de los tópicos en el corpus obtenida a partir de la matriz M_2 y KL es la divergencia simétrica de Kullback-Leibler definida como:

$$KL(\bar{X}, \bar{Y}) = \sum_{i=1}^{n} x_i \cdot \log\left(\frac{x_i}{y_i}\right) + \sum_{i=1}^{n} y_i \cdot \log\left(\frac{y_i}{x_i}\right)$$

Valores pequeños de (2.31) son evidencia de un mejor modelo.

2.6.5.4. Medida basada en la divergencia de Jensen-Shannon (Romain Deveaud)

Partiendo de las métricas anteriores Romain Deveaud et al. proponen en 2014 [38] una medida basada en la noción de cercanía entre tópicos usando la divergencia de Jensen-Shannon.

Suponga que se tiene un modelo de asignación Dirichlet latente con k tópicos. Sean T_k el conjunto de tópicos inducido por este modelo y W_z el conjunto de las n palabras más probables en el tópico $z \in \mathrm{T}_k$. Bajo este esquema se define el número óptimo de tópicos como:

$$\hat{k} = \arg\max_k \frac{1}{k(k-1)} \sum_{\mathrm{T}_k} JS\left(z\,|z'\right) \tag{2.32}$$

donde $JS(z\,|z')$ es la divergencia de Jensen-Shannon definida como:

$$JS\left(z\,|z'\right) = \frac{1}{2}\left[\sum_{w \in W_z \cap W_{z'}} P(w|z) \log \frac{P(w|z)}{P(w|z')} + \sum_{w \in W_z \cap W_{z'}} P\left(w\,|z'\right) \log \frac{P(w|z')}{P(w|z)}\right]$$

Gráficamente se buscará el valor de k que maximice la curva inducida por (2.32).

2.6.5.5. Log-verosimilitud

La log-verosimilitud es uno de los métodos más comunes para evaluar la bondad de ajuste de un modelo probabilístico y en particular es útil para encontrar el número de tópicos. Usando inferencia variacional es posible derivar una estimación de la verosimilitud a partir de una cota inferior obtenida a través de este método [13]. Usando el algoritmo de Gibbs la log-verosimilitud puede ser obtenida aplicando logaritmo a la ecuación (2.20):

$$\log P(w|z) = k \log\left(\frac{\Gamma(t\beta)}{\Gamma(\beta)^t}\right) + \sum_{l=1}^{k}\left(\sum_{w=1}^{t} \log\Gamma\left(n_l^{(w)} + \beta\right) - \log\Gamma\left(n_l^{(\cdot)} + t\beta\right)\right)$$

2.6.5.6. Perplejidad

Otros métodos de selección de tópicos pueden ser implementados particionando el corpus en una muestra de entrenamiento y una muestra de prueba. La perplejidad es una medida de la calidad del modelo ampliamente usada en modelación de lenguaje, tiene la propiedad de ser decreciente en la verosimilitud de la muestra de prueba y es equivalente a la media geométrica inversa de la verosimilitud por término [13]. La perplejidad en la muestra de prueba D_{prueba} está definida como:

$$perplejidad\left(D_{prueba}\right) = \exp\left\{-\frac{\sum_{i=1}^{n}\sum_{j=1}^{t_i} \log P\left(w_{i,j}\big|\,\text{modelo}\right)}{\sum_{i=1}^{n} t_i}\right\} \tag{2.33}$$

Valores pequeños de (2.33) son evidencia de un mejor ajuste en la muestra de prueba y por lo tanto una buena generalización del modelo. Los valores de las probabilidades $P\left(w_{i,j}\middle|\text{modelo}\right)$ no pueden ser evaluados directamente y se tiene que recurrir a estimaciones. Wallach et al. [39] propone una serie de estimaciones basadas en técnicas de muestreo por importancia y medias armónicas.

2.6.6. Reducción de la dimensión

Los modelos tópicos aquí presentados pueden ser usados como métodos de reducción de la dimensión al cambiar el espacio original de los documentos por un espacio reducido dado por las densidades posteriores de las distribuciones Dirichlet. En particular las ecuaciones (2.22) pueden ser usadas para obtener representaciones de los documentos y las palabras en $k-1$ dimensiones. Si se piensa en las frecuencias normalizadas de los documentos como su representación original se puede hablar de un cambio dimensional del simplejo de las palabras en $t-1$ dimensiones al simplejo de los tópicos en $k-1$ dimensiones [13]. Como se ha comentado anteriormente este conveniente cambio de dimensión permite reducir los problemas asociados a la polisemia y la sinonimia, esto en particular es útil cuando se desea implementar un algoritmo de clasificación ya que permite incrementar su precisión y eficientar el proceso de entrenamineto. Blei et al. (2003) [13] experimentan con una versión optimizada de las máquinas de soporte vectorial (SVMlight) en un problema de clasificación de texto obteniendo resultados satisfactorios cuando se aplica el algoritmo en la dimensión reducida en vez de la dimensión original.

2.7. Nota bibliográfica

El enfoque aquí presentado es el que comúnmente se adopta en la mayoría de la literatura y en particular se puede encontrar una amplia introducción a estos temas en [5]. Las técnicas de preprocesamiento y representación de texto conforman las discusiones típicas sobre el tema y pueden ser encontradas en [2] y [23]. En [3] se puede encontrar la exposición sobre medidas de similaridad para texto desarrollada en la sección 2.3, así como una completa discusión del tema para diferentes contextos de la minería de datos.

La teoría presentada sobre reducción de la dimensión y modelación de tópicos latentes está basada principalmente en una serie de resultados de investigación sobre modelación de tópicos publicados por David M. Blei, Andrew Y. Ng, Michael I. Jordan, John D. Lafferty y Jon D. McAuliffe (véase [13], [32] y [34]), quienes son los principales artífices de estos métodos. En partícular en [28], [2] y [25] pueden encontrarse capítulos completos dedicados a la exposición teórica de este tipo de modelos. Los algoritmos más fundamentales del análisis semántico latente, presentados en las dos primeras secciones son atribuidos a las investigaciones de Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Richard Harshman y Thomas Hofmann (véase [40], [41]), que originalmente enfocaron estos métodos al área de la recuperación de la información para resolver problemas de consulta e indexamiento.

Capítulo 3

Técnicas para clasificación empleadas en minería de textos

3.1. Selección de variables

Antes de aplicar cualquier técnica de clasificación es necesario seleccionar las características más importantes en la base de datos con el fin de guiar al algoritmo hacia un resultado favorable. Una adecuada selección de características o variables puede eliminar el ruido presente entre las clases a la vez que reduce la dimensión de la base para eficientar el proceso de aprendizaje. Bajo el contexto de la minería de texto las características que nos interesan seleccionar son aquellas palabras que mejor discriminen las clases a las que pertenecen los documentos. Esta tarea es llevada a cabo primero bajo un enfoque no supervisado como el de la sección 2.1 en el que las *palabras vacías* son removidas y las palabras similares son sintetizadas en una sola raíz (*stemming*). Un segundo enfoque opta por aprovechar la información de las clases y supervisar el proceso de selección usando diferentes medidas de asociación entre palabras y clases. En esta sección se describen algunas de las técnicas más populares de selección supervisada de caraterísticas para clasificación de texto.

3.1.1. Índice de Gini

El índice de Gini es uno de los métodos de selección de variables más usados en minería de textos. Este método consiste en asignar valores crecientes a las palabras en el léxico según su capacidad discriminativa. Sean $p_1(w),...,p_k(w)$ las probabilidades condicionales de observar el documento d en la clase i dado que este contiene la palabra w, es decir $p_i(w) = P(d \in i | w \subset d)$ para $i = 1,...,k$. Así, $p_i(w)$ podría ser por ejemplo la probabilidad de que una revista pertenezca a la clase "deportes" dado que contiene la palabra "football". Según lo anterior se tiene que $\sum_i p_i(w) = 1$ y se define entonces el *índice de Gini* para la palabra "w" como:

$$G(w) = \sum_{i=1}^{k} p_i(w)^2$$

Observe que en el peor de los casos la palabra w aparece en todos los documentos y entonces $G(w) = 1/k$, en caso contrario solo aparece en una clase de documentos y entonces $G(w) = 1$. Así, el rango del ínidice de Gini es $(1/k, 1)$. Geométricamente el índice de Gini mide la norma de los vectores de la forma $(p_1(w),...,p_k(w))$ que pertenecen al espacio simplejo de k dimensiones (i.e. satisfacen que $\sum_{i=1}^{k} p_i(w) = 1$). Note que la norma del vector $(p_1(w),...,p_k(w))$ se maximiza cuando exactamente una de sus entradas vale uno y el resto cero y que esto pasa cuando el vector coincide con uno de los vértices del simplejo. Así, entre más cerca se encuentre el vector $(p_1(w),...,p_k(w))$ de algún vértice, mayor será la capacidad discriminativa de la palabra w.

En casi todos los casos la muestra de documentos estará desbalanceada en mayor o menor grado (i.e. se tendrá un número diferente de observaciones en cada clase) y será necesario estandarizar el índice de Gini para considerar este efecto en el cálculo de las probabilidades como:

$$p_i{}'(w) = \frac{p_i(w)/P_i}{\sum_{j=1}^{k} p_j(w)/P_j}$$

donde P_i representa la probabilidad de que un documento pertenezca a la i-ésima clase: $P_i = P(d \in i)$. Observe que cuando la muestra está balanceada (se tiene el mismo número de documentos por clase), se obtiene $p_i{}'(w) = p_i(w)$. El índice de Gini para muestras no balanceadas se calcula finalmente como:

$$G(w) = \sum_{i=1}^{k} p_i{}'(w)^2$$

3.1.2. Ganancia de información

El criterio de *ganancia de información* es ampliamente usado en al área del aprendizaje automatizado para evaluar la bondad de discriminación para cada término. Entre mayor sea su valor mayor será la capacidad discriminatoria de la palabra. Sea $F(w)$ la probabilidad de que la palabra w esté contenida en algún documento: $F(w) = P(w \subset d)$, se define el criterio de ganancia de información para el término "w" como:

$$\begin{aligned} I(w) &= -\sum_{i=1}^{k} P_i \cdot \log(P_i) + F(w) \cdot \sum_{i=1}^{k} p_i(w) \cdot \log(p_i(w)) \\ &+ (1 - F(w)) \cdot \sum_{i=1}^{k} (1 - p_i(w)) \cdot \log(1 - p_i(w)) \end{aligned}$$

Así, para cada término en el léxico se calcula la cantidad $I(w)$ estimando las probabilidades correspondientes a partir de la muestra y se eliminan aquellas palabras para las cuales el criterio toma un valor menor que cierto límite preestablecido.

3.1.3. Información mutua

El criterio de información mutua se deriva de la teoría de la información y mide la información aportada cuando el término w y la clase i ocurren al mismo tiempo:

$$M_i(w) = \log\left(\frac{F(w) \cdot p_i(w)}{F(w) \cdot P_i}\right) = \log\left(\frac{p_i(w)}{P_i}\right)$$

Note que el numerador "$F(w) \cdot p_i(w) = P(d \in i, w \subset d)$" denota la probabilidad conjunta entre el término w y la clase i mientras que el denominador "$F(w) \cdot P_i$" asume independencia para dicha probabilidad. Valores positivos o negativos de $M_i(w)$ indican correlación entre el término y la clase mientras que valores cercanos a cero son evidencia de poca asociación. En general el criterio de información mutua está definido para una clase en particular, por lo que será necesario considerar el promedio o el máximo sobre las clases:

$$M_{prom}(w) = \sum_{i=1}^{k} P_i \cdot M_i(w) \tag{3.1}$$

$$M_{\text{máx}}(w) = \max_i \{M_i(w)\} \tag{3.2}$$

3.1.4. Estadístico Ji-cuadrada

El estadístico Ji-cuadrada ofrece una manera alternativa de medir la ausencia de independencia entre términos y clases. Sea d el número de documentos en el corpus, se define el estadístico Ji-cuadrada entre el término w y la clase i como:

$$\chi_i^2(w) = \frac{n \cdot F(w)^2 \cdot (p_i(w) - P_i)^2}{F(w) \cdot (1 - F(w)) \cdot P_i \cdot (1 - P_i)}$$

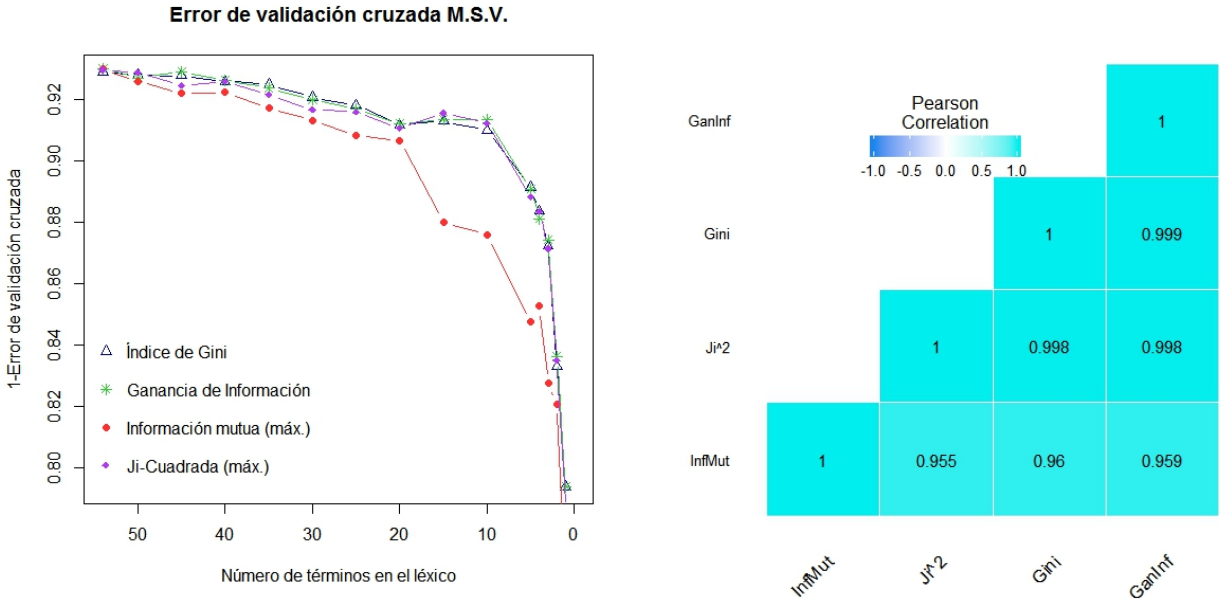


Figura 3.1: A la izquierda se muestra el desempeño de los métodos de selección de variables. Se aplicó una máquina de soporte vectorial a la base de datos SPAM y se calculó el error de validación cruzada como medida para cuantificar la exactitud en las predicciones. A la derecha se muestra la matriz de correlación de Pearson de los resultados obtenidos.

Al igual que en el caso anterior, el estadístico χ^2 toma el valor de cero cuando hay independencia entre clases y términos. Finalmente se promedia o maximiza el valor del estadístico sobre todas las posibles categorías de documentos:

$$\chi^2_{prom}(w) = \sum_{i=1}^{k} P_i \cdot \chi^2_i(w) \tag{3.3}$$

$$\chi^2_{\text{máx}}(w) = \max_i \left\{ \chi^2_i(w) \right\} \tag{3.4}$$

Es posible medir el desempeño de los métodos de selección de variables considerando una base de datos en particular, una técnica de clasificación y una medida de error, sin embargo el costo computacional de tal experimento puede ser alto y por ello se recomienda elegir un algoritmo cuyo tiempo de entrenamiento sea corto. En este caso se ha implementado dicho experimento entrenando una máquina de soporte vectorial (ver sección 3.3) en cada subconjunto de variables y midiendo su error de validación cruzada para cada método. Los resultados obtenidos se muestran en la figura 3.1. A partir de estos resultados es posible notar que es factible reducir en un 81.48 % la dimensión de los datos sacrificando apenas un 2.68 % de precisión promedio en el error de predicción si se consideran el índice de Gini, la medida de ganancia de información y el estadístico Ji-cuadrada como métodos de selección de variables. El método de información mutua presenta un bajo rendimiento en comparación con el resto de los métodos. La matriz de correlación de Pearson [1] de la figura 3.1 también muestra que existe una fuerte relación lineal positiva entre los métodos de selección de variables.

3.2. Naive Bayes

Los clasificadores de la familia Naive Bayes se caracterizan por su simplicidad y su buen desempeño aún en problemas de alta dimensionalidad. Este enfoque permite incorporar además información adicional como la que se puede encontrar en páginas de internet como blogs o redes sociales y mejorar con ello la precisión del algoritmo. El modelo supone independencia condicional entre términos de la misma clase y por ello recibe

[1] El coeficiente de correlación de Pearson es una medida de la relación lineal entre dos variables aleatorias y toma valores entre -1 y 1. Valores cercanos a uno indican una correlación positiva y valores cercanos a menos uno una correlación negativa entre las variables.

el adjetivo de "Naive" o "ingenuo". El supuesto de independencia claramente no se cumple en este contexto, ya que la ocurrencia de una palabra generalmente condiciona la ocurrencia de otra en la mayoría de los casos. A pesar de que asumir independencia parece ser bastante restrictivo los resultados obtenidos con este tipo de clasificadores son más que aceptables y por ello ampliamente recurridos en la práctica. A continuación se describen dos de los esquemas fundamentales de este tipo de clasificadores.

3.2.1. Modelo Bernoulli multivariado

El modelo Bernoulli multivariado considera sólo la presencia o ausencia de cada término en el documento sin importar la frecuencia con que ésta se presenta, así por ejemplo un documento de un léxico de cinco palabras en el que sólo la primera y la cuarta se encuentran presentes puede ser respresentado como $\bar{X} = (1,0,0,1,0)$. A continuación se describe el modelo general bajo esta representación. Sea $\bar{X} = (x_1,...,x_d)$ un nuevo documento que no ha sido clasificado, el objetivo principal bajo un enfoque Bayesiano es modelar la densidad posterior $P(C = c\,|X_1 = x_1,...,X_d = x_d)$ a partir de la muestra de entrenamiento y elegir aquella clase que la maximize. Usando el Teorema de Bayes se puede calcular esta densidad posterior de la siguiente manera:

$$\begin{aligned} P(C = c\,|X_1 = x_1,...,X_d = x_d) &= \frac{P(C = c)\,P(X_1 = x_1,...,X_d = x_d\,|C = c)}{P(X_1 = x_1,...,X_d = x_d)} \\ &\propto P(C = c)\,P(X_1 = x_1,...,X_d = x_d\,|C = c) \\ &\approx P(C = c)\prod_{i=1}^{d} P(X_i = x_i\,|C = c) \end{aligned}$$

Observe que en la segunda línea de la ecuación anterior se ha omitido el denominador ya que éste no depende de la clase. La última línea es posible por el supuesto "naive" de independencia condicional entre términos. Las probabilidades $P(X_i = x_i\,|C = c)$ para $i = 1,...,d$ y $P(C = c)$ son estimadas directamente de la muestra de entrenamiento considerando solamente si la palabra se encuentra o no en el documento sin tomar en cuenta así la frecuencia con la que aparece. La clase seleccionada por el algoritmo es finalmente la que maximiza la densidad posterior:

$$\begin{aligned} \hat{c} &= \arg\max_i P(C = c\,|X_1 = x_1,...,X_d = x_d) \\ &= \arg\max_i P(C = c)\prod_{i=1}^{d} P(X_i = x_i\,|C = c) \end{aligned}$$

Es importante notar que dada la escasez de la matriz documento-término el modelo Bernoulli dará mayor importancia a la ausencia que a la presencia de una palabra, lo cual resulta inconveniente ya que en la mayoría de los casos la presencia de una palabra en un documento puede determinar de manera más efectiva la clase a la que éste pertenece. La frecuencia con la que se presentan las palabras tampoco es tomada en cuenta bajo este enfoque, por lo que en ciertos casos será necesario usar otro modelo que considere esta característica.

3.2.2. Modelo multinomial

A diferencia del método anterior, el modelo multinomial permite considerar la frecuencia con que se presentan las palabras en el documento. Así, por ejemplo, un documento cuyo vector de características está dado por $\bar{X} = (5,0,0,3,1)$ contiene cinco veces la primera palabra, tres veces la cuarta y una vez la quinta. El objetivo nuevamente es estimar la densidad posterior $P(C = c\,|X_1 = x_1,...,X_d = x_d)$, pero en este caso se asume una densidad a priori multinomial para la distribución de la frecuencia en los documentos:

$$P(X_1 = x_1,...,X_d = x_d\,|C = c) \approx \binom{n}{x_1...x_d} \prod_{i=1}^{k} p_c(i)^{x_i}$$

donde $n = \sum_{i=1}^{d} x_i$ denota el número total de palabras contenidas en el documento y $p_c(i)$ la probabilidad estimada de observar la i-ésima palabra en la clase "c". La densidad posterior bajo estos supuestos es calculada entonces de la siguiente manera:

$$\begin{aligned} P(C=c\,|X_1=x_1,...,X_d=x_d) &\propto P(C=c)\cdot P(X_1=x_1,...,X_d=x_d\,|C=c) \\ &\approx P(C=c)\cdot \left(\begin{array}{c} n \\ x_1...x_d \end{array}\right)\prod_{i=1}^{k} p_c(i)^{x_i} \\ &\propto P(C=c)\cdot \prod_{i=1}^{k} p_c(i)^{x_i} \end{aligned} \tag{3.5}$$

La última igualdad de la expresión anterior es proporcional ya que el factor de normalización $\left(\begin{array}{c} n \\ x_1...x_d \end{array}\right)$ no depende de la clase. La solución definida por el modelo multinomial es finalmente:

$$\begin{aligned} \hat{c} &= \arg\max_i P(C=c\,|X_1=x_1,...,X_d=x_d) \\ &= \arg\max_i P(C=c)\cdot \prod_{i=1}^{k} p_c(i)^{x_i} \end{aligned} \tag{3.6}$$

Algunas investigaciones muestran que el modelo Bernoulli puede dar mejores resultados cuando se considera un léxico pequeño, mientras que el modelo multinomial funciona mejor con léxicos grandes y supera incluso al modelo Bernoulli si el tamaño de vocabulario es elegido apropiadamente con un 27 % de reducción promedio en el error según [2].

3.3. Máquinas de soporte vectorial

Las máquinas de soporte vectorial son uno de los algoritmos más estudiados dentro del aprendizaje automatizado. En los últimos años han cobrado gran importancia debido a su eficiencia computacional y su precisión en tareas que implican trabajar con grandes cantidades de datos. Algunos de los problemas de aplicación incluyen clasificación, regresión, detección de novedades y ranqueo. A continuación se desarrolla el fundamento teórico de las máquinas de soporte vectorial para el caso en el que los conjuntos son linealmente separables.

Dada una muestra de entrenamiento $S = \{(x,y) : x \in \Re^n, y \in \{-1,+1\}\}$, donde x representa un conjunto de características o covariables, y y es una variable indicadora que identifica a qué grupo o clase pertenece la observación. El objetivo de la clasificación lineal es encontrar algún hiperplano que discrimine ambos grupos. Partiendo del hiperplano canónico, definido como aquel tal que $\min_{(x,y)\in S} |w\cdot x+b| = 1$, se define el margen de separación lineal como:

$$\rho = \min_{(x,y)\in S} \frac{|w\cdot x+b|}{\|w\|} = \frac{\min_{(x,y)\in S} |w\cdot x+b|}{\|w\|} = \frac{1}{\|w\|}$$

La solución obtenida por el algoritmo es el hiperplano que maximiza el margen ρ a la vez que clasifica correctamente los puntos. Esto se puede plantear como el siguiente problema de optimización convexa:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \tag{3.7}$$

$$\text{sujeto a} : y_i(w\cdot x_i+b) \geq 1, \forall i \in [1,m]$$

Se establece ahora al lagrangiano asociado a este problema de la siguiente manera:

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i [y_i(w\cdot x_i+b)-1] \tag{3.8}$$

Del lagrangiano anterior y aplicando las condiciones del Teorema de Karush-Kuhn-Tucker [42] se obtienen las siguientes expresiones:

$$\begin{aligned}\frac{\delta L}{\delta w} &= w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0 \Rightarrow w = \sum_{i=1}^{m} \alpha_i y_i x_i \\ \frac{\delta L}{\delta b} &= -\sum_{i=1}^{m} \alpha_i y_i = 0 \Rightarrow \sum_{i=1}^{m} \alpha_i y_i = 0\end{aligned} \tag{3.9}$$

$$\forall i, \alpha_i \left[y_i \left(w \cdot x_i + b \right) - 1 \right] = 0 \Rightarrow \alpha_i = 0 \vee y_i \left(w \cdot x_i + b \right) = 1 \tag{3.10}$$

De las ecuaciones (3.9) se puede apreciar que la solución sólo dependerá de aquellos vectores x_i tales que $\alpha_i \neq 0$. Los vectores x_i con esta propiedad son llamados vectores de soporte. De la ecuación (3.10), si $\alpha_i \neq 0$ entonces $y_i \left(w \cdot x_i + b \right) = 1$. Esto significa que los vectores de soporte se encuentran en los hiperplanos marginales $w \cdot x_i + b = \pm 1$. Estas últimas observaciones justifican el nombre del algoritmo y la figura 3.2 ejemplifica lo dicho hasta este punto.

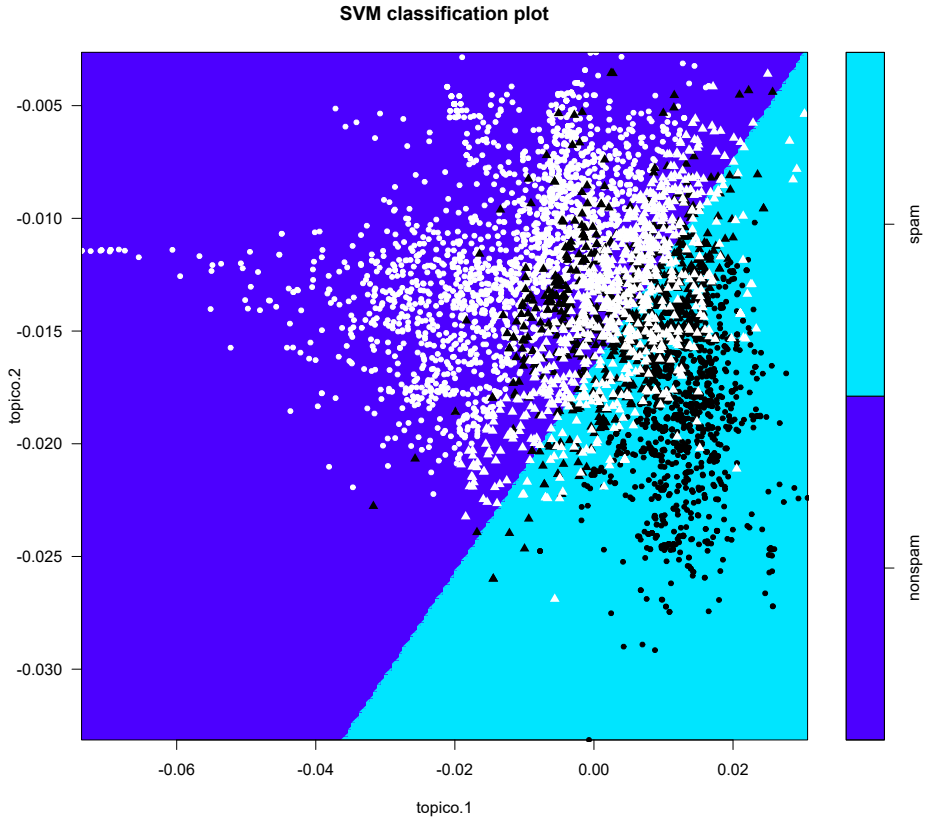


Figura 3.2: Solución definida por una máquina de soporte vectorial para la base de datos SPAM. La proyección se obtuvo usando la descomposición espectral del análisis semántico latente en dos dimensiones. Las figuras triangulares representan los vectores de soporte.

A diferencia de otros algoritmos de clasificación lineal, las máquinas de soporte vectorial pueden definir fronteras no lineales de manera indirecta introduciendo el uso de funciones kernel. Esto es posible en general para cualquier algoritmo cuya solución dependa de las covariables x_i , $i = 1, \ldots, m$ sólo a través de los productos punto $x_i \cdot x_j$: $i, j = 1, \ldots, m$. Para ver que la solución de las máquinas de soporte vectorial depende sólo de productos punto basta con escribir el problema de optimización convexa (3.7) de manera equivalente en su forma dual:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \left(x_i \cdot x_j \right) \tag{3.11}$$

$$\text{sujeto a} : \alpha_i \geq 0 \wedge \sum_{i=1}^{m} \alpha_i y_i = 0, \forall i \in [1, m]$$

El problema dual (3.11) se obtiene de sustituir las ecuaciones (3.9) en el lagrangiano (3.8) y simplificar. Los problemas (3.7) y (3.11) se resuelven usando métodos numéricos y muchos de ellos ya están optimizados para resolver este problema en particular. A continuación se hace una importante observación que motiva el uso de funciones kernel para definir fronteras no lineales. Según las ecuaciones (3.9) y usando el hecho de que cualquier vector de soporte x_j satisface $w \cdot x_j + b = y_j$, la hipótesis seleccionada por el algoritmo puede expresarse de la siguiente manera:

$$\begin{aligned} h(x) &= \operatorname{sgn}(w \cdot x + b) = \operatorname{sgn}(w \cdot x + (y_i - w \cdot x_i)) && (3.12) \\ &= \operatorname{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i (x_i \cdot x) + \left(y_j - \sum_{i=1}^{m} \alpha_i y_i (x_i \cdot x_j)\right)\right) && (3.13) \end{aligned}$$

Esto significa que las soluciones generadas por las máquinas de soporte vectorial dependen sólo de productos punto y no de los vectores mismos, hecho que podrá ser aprovechado en el planteamiento general del algoritmo cuando se tengan fronteras no lineales.

En este punto es natural preguntarse: ¿qué ocurre cuando no existe ningún hiperplano que separe perfectamente los puntos? Para resolver esta cuestión basta con introducir variables de holgura a los modelos (3.7) y (3.11) de tal manera que se relajen las restricciones y se puedan obtener versiones más generalizadas de ellos:

$$\min_{w,b,\varepsilon} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \varepsilon_i{}^p \tag{3.14}$$

$$\text{sujeto a} : y_i (w \cdot x_i + b) \geq 1 - \varepsilon_i \wedge \varepsilon_i \geq 0, \forall i \in [1, m]$$

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \tag{3.15}$$

$$\text{sujeto a} : 0 \leq \alpha_i \leq C \wedge \sum_{i=1}^{m} \alpha_i y_i = 0, \forall i \in [1, m]$$

En este caso las variables $\varepsilon_i, i = 1, ..., m$ se introducen al modelo para relajar las restricciones y se asigna un costo de pérdida $C \geq 0$ así como una función de pérdida inducida por la variable p. Las elecciones más comunes de p son 1 y 2 y en este trabajo asumiremos $p = 1$. En general C es un parámetro libre en el modelo y se determina por validación cruzada. En este caso los vectores de soporte serán aquellos que se encuentren en los hiperplanos marginales o bien sean valores atípicos en el modelo, entendiéndose por valores atípicos aquellos x_i para los cuales $\varepsilon_i > 0$.

Es posible obtener una generalización del algoritmo mediante el uso de funciones kernel. Una función kernel K sobre X es aquella tal que $K : X \times X \to \Re$. La idea es seleccionar una función kernel tal que induzca un producto punto en un espacio de mayor dimensión en el que los conjuntos sean linealmente separables. Introduciendo funciones kernel generalizamos ahora el problema de optimización convexa (3.15) y la hipótesis seleccionada por el algoritmo de la siguiente manera:

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{3.16}$$

$$\text{sujeto a} : 0 \leq \alpha_i \leq C \wedge \sum_{i=1}^{m} \alpha_i y_i = 0, \forall i \in [1, m]$$

$$h(x) = \operatorname{sgn}(w \cdot x + b) = \operatorname{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i K(x_i, x) + b\right) \tag{3.17}$$

Para que el problema (3.16) tenga solución es necesario pedir que la función kernel elegida satisfaga la condición de Mercer [43]: "El kernel K debe ser simétrico definido positivo". Esto último significa que para cualquier subconjunto $\{x_1,...,x_m\} \subseteq X$, la matriz $K = [K(x_i,x_j)]_{ij} \in \Re^{m\times m}$ es simétrica y semidefinida positiva. Algunas de las elecciones más comúnes para la función kernel son las siguientes:

- **Kernel Gaussiano o radial**
  Sea $\sigma > 0$, un kernel Gaussiano o radial es aquel definido $\forall x,y \in \Re^n$ como:

$$K(x,y) = \exp\left(-\sigma\|x-y\|^2\right)$$

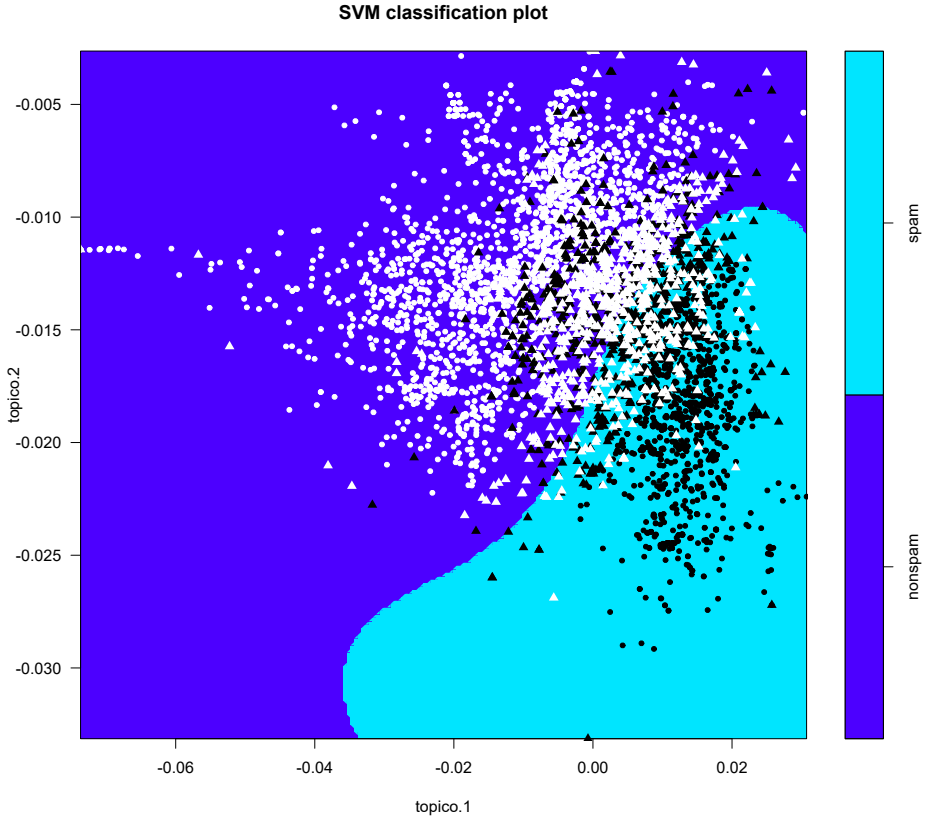


Figura 3.3: Solución definida por una máquina de soporte vectorial para la base de datos SPAM con un kernel Gaussiano o radial. Se han usado los parámetros por defecto definidos para la clase SVM del paquete e1071 de R.

- **Kernel polinomial**
  Sean $b,c > 0$, un kernel polinomial de grado $d \in \mathbb{N}$ queda definido $\forall x,y \in \Re^n$ de la siguiente manera:

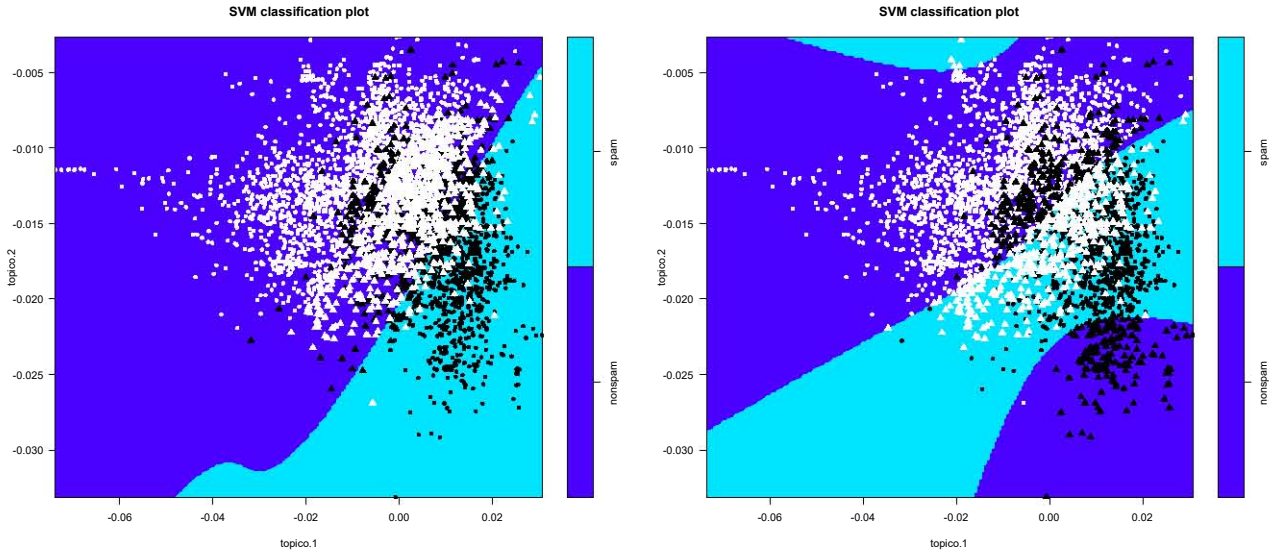
$$K(x,y) = (b(x\cdot y)+c)^d$$

- **Kernel sigmoidal**
  Sean $a > 0$,$b \geqslant 0$ constantes, el kernel sigmoide se define $\forall x,y \in \Re^n$ como:

$$K(x,y) = \tanh(a(x\cdot y)+b)$$

3.4. Máquinas de relevancia vectorial

Las máquinas de relevancia vectorial (MRV) pueden ser consideradas como una versión Bayesiana de las máquinas de soporte vectorial (MSV). Fueron desarrolladas por Michael E. Tipping como resultado de



(a) Kernel polinomial

(b) Kernel sigmoide

Figura 3.4: Solución definida por una máquina de soporte vectorial para la base de datos SPAM con un kernel polinomial (izquierda) y uno sigmoide (derecha). Se han usado los parámetros por defecto definidos para la clase svm de la biblioteca e1071.

sus investigaciones en Microsoft Research en Cambridge (UK) durante 1998-2006 [44]. Este modelo se ha popularizado no sólo gracias a sus similitudes con las ya conocidas máquinas de soporte vectorial sino a las ventajas que presenta frente a éstas. Entre algunas de sus características principales destacan las siguientes:

- A diferencia de las MSV, en las MRV no hay parámetros libres que estimar.
- El algoritmo es robusto dado que opera con un número escaso de vectores de relevancia.
- La precisión de las MRV es equiparable a la obtenida con las MSV
- Se puede tener una medida de la incertidumbre dado que las salidas del algoritmo son probabilísticas.

Dada la hipótesis seleccionada por las máquinas de soporte vectorial el algoritmo comienza por estimar los pesos $w_i = a_i y_i$, para $i = 1,...,n$ así como el intercepto $w_0 = b$. El modelo (3.17) escrito con esta notación tiene entonces la forma:

$$y(x) = \sum_{i=1}^{m} w_i K(x_i, x) + w_0 \tag{3.18}$$

Al igual que en el caso de una regresión lineal, asumiremos un error aleatorio $\varepsilon_i \sim N\left(0, \sigma^2\right)$ y dado un conjunto de entrenamiento $\{x_i, t_i\}_{i=1}^{m}$, supondremos que nuestro modelo es representativo al considerar dicho error aleatorio. De esta manera, el modelo toma la forma $t_i(x) = y_i(x) + \varepsilon_i$. Así, bajo estos supuestos, tenemos que $p(t|x) \sim N\left(y_i(x), \sigma^2\right)$. La verosimilitud de la muestra de entrenamiento dados los parámetros desconocidos queda entonces expresada como:

$$P\left(t \left| w, \sigma^2 \right.\right) = \prod_{i=1}^{m} N\left(y_i(x), \sigma^2\right) = \left(2\pi\sigma^2\right)^{-\frac{m}{2}} \exp\left\{-\frac{1}{2\sigma^2}\|t - \Phi w\|^2\right\}$$

de donde $t = (t_1,...,t_m)$, $w = (w_0,...,w_m)$ y Φ es la matriz de funciones kernel definida como: $\Phi_{ij} = K(x_i, x_{j-1})$ y $\Phi_{i1} = 1$. Para proceder a hacer inferencia Bayesiana, asumimos una distribución inicial sobre el parámetro $w = (w_0,...,w_m)$ de la siguiente manera:

$$p(w|\alpha) = \prod_{i=1}^{m} N\left(w_i \left| 0, \alpha_i^{-1}\right.\right)$$

Note que de esta manera cada w_i seguirá una distribución normal de media cero parametrizada por el parámetro de precisión α_i^{-1}. Esta elección distribucional permite que el modelo sea escaso y en consecuencia robusto, además se logra flexibilidad gracias a las precisiones α_i^{-1} asociadas a cada w_i. Se busca entonces maximizar la distribución posterior de los parámetros desconocidos dada la muestra: $P\left(w,\alpha,\sigma^2\,|t\right)$. Dicha distribución puede descomponerse usando probabilidad condicional como:

$$P\left(w,\alpha,\sigma^2\,|t\right) = P\left(w\,|t,\alpha,\sigma^2\right)P\left(\alpha,\sigma^2\,|t\right) \tag{3.19}$$

donde el primer miembro de la ecuación (3.19) tiene la distribución:

$$P\left(w\,|t,\alpha,\beta\right) \sim N\left(\mu,\Sigma\right)$$

$$\begin{aligned} \mu &= \beta\Sigma\Phi^T t \\ \Sigma &= \left(A+\beta\Phi^T\Phi\right)^{-1} \\ A &= diag\left(\alpha\right) \end{aligned} \tag{3.20}$$

Con el fin de evaluar los parámetros μ y Σ se buscan α y β que maximicen el segundo miembro de la ecuación (3.19), el cual puede ser descompuesto como:

$$P\left(\alpha,\sigma^2\,|t\right) \propto P\left(t\,|\alpha,\sigma^2\right)P\left(\alpha\right)P\left(\sigma^2\right) \tag{3.21}$$

Bajo el supuesto de uniformidad en las distribuciones iniciales nos enfocamos a maximizar la verosimilitud marginal. Sea $\beta^{-1}=\sigma^2$, entonces:

$$P\left(t\,|\alpha,\sigma^2\right) = P\left(t\,|\alpha,\beta\right) = \int P\left(t\,|w,\beta\right)P\left(w\,|\alpha\right)dw \tag{3.22}$$

$$= \left(\frac{\beta}{2\pi}\right)^{\frac{m}{2}}\left(\frac{1}{2\pi}\right)^{\frac{n}{2}}\prod_{i=1}^{n}\alpha_i^{\frac{1}{2}}\exp\left\{-E\left(t\right)\right\}\left(2\pi\right)^{\frac{n}{2}}\left|\Sigma\right|^{\frac{1}{2}} \tag{3.23}$$

donde

$$E\left(t\right) = \frac{1}{2}\left(\beta t^T t - m^T\Sigma^{-1}m\right)$$

Aplicando la función logaritmo obtenemos la log verosimilitud marginal:

$$\log P\left(t\,|\alpha,\sigma^2\right) = \frac{m}{2}\log\beta - E\left(t\right) - \frac{1}{2}\log\left|\Sigma\right| - \frac{m}{2}\log\left(2\pi\right) + \frac{1}{2}\sum_{i=1}^{n}\log\alpha_i \tag{3.24}$$

Finalmente se maximiza la función (3.24) usando un procedimiento conocido como aproximación de evidencia. Este procedimiento se describe a continuación. Derivando parcialmente para cada α_i e igualando a cero y despejando obtenemos:

$$\alpha_i^{nuevo} = \frac{1-\alpha_i\Sigma_{ii}}{\mu_i^2} = \frac{\gamma_i}{\mu_i^2} \tag{3.25}$$

$$\beta = \frac{m-\sum_i \gamma_i}{\left\|t-\Phi\mu\right\|^2} \tag{3.26}$$

El algoritmo de aproximación de la evidencia procede entonces de la siguiente manera:

i) Seleccionar valores iniciales para α_i y β.

ii) Calcular μ y Σ de las ecuaciones (3.20).

iii) Actualizar α_i y β usando (3.26).

iv) Verificar los criterios de convergencia o divergencia para los parámetros. Sólo aquellos que convergan serán los vectores de relevancia.

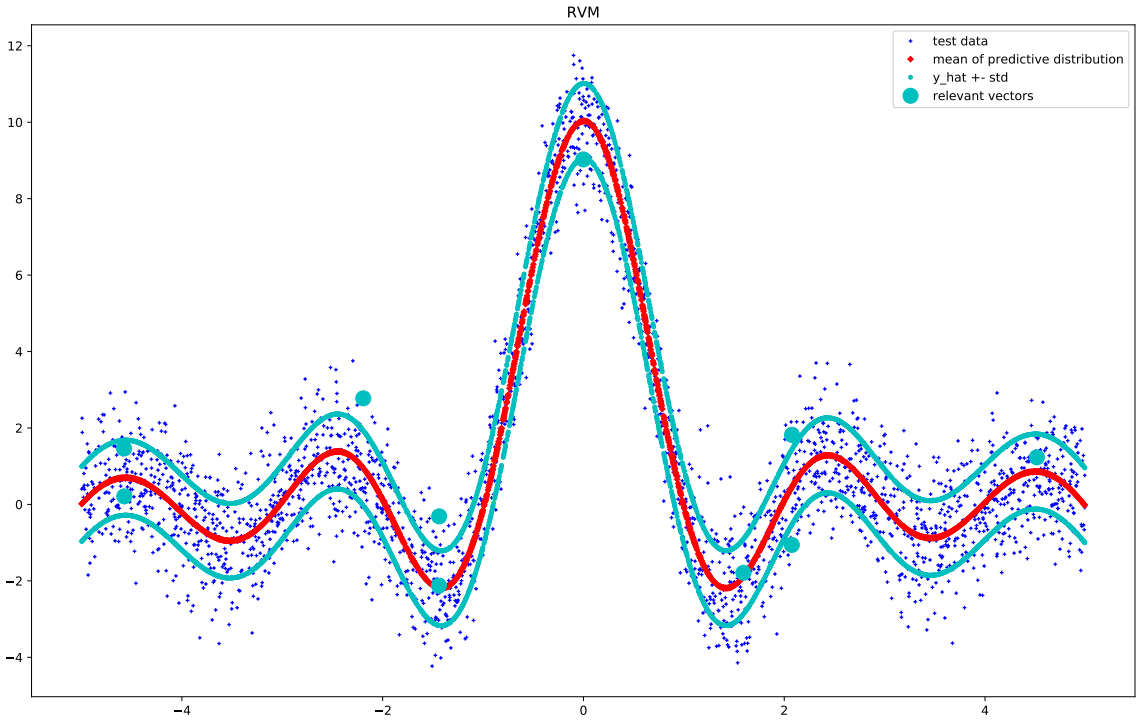


Figura 3.5: Ajuste obtenido por una regresión de relevancia vectorial. La muestra de entrenamiento proviene de una función seno con un ruido Gaussiano [45].

v) En caso de no cumplirse la convergencia en *iv*) repetir los pasos anteriores.

El modelo lineal anteriormente descrito es conocido como regresión de relevancia vectorial y puede ser extendido al caso de la clasificación binaria como un modelo lineal generalizado introduciendo la función logística sigmoide $\sigma(y) = 1/(1+e^{-y})$ en la función de verosimilitud:

$$P(t|w) = \prod_{n=1}^{N} \sigma\{y(x_n)\}^{t_n} [1 - \sigma\{y(x_n)\}]^{1-t_n}$$

Esta última verosimilitud puede ser maximizada por procedimientos similares a los anteriores según Mackay [46].

La figura 3.6 muestra los resultados del modelo ajustado para el caso de la clasificación binaria. Nótese que en este caso la función que se predice es una variable dicotómica y que el modelo es, en efecto, escaso dado que se tienen tres vectores de relevancia (se muestran en azul).

3.5. Redes neuronales

Las redes neuronales son el resultado de más de cincuenta años de investigación motivada por el deseo de desarrollar un modelo matemático que se asemeje en capacidades y estructura al funcionamiento del cerebro humano. Aunque ahora constituyen uno de los modelos de clasificación y regresión más populares en el aprendizaje automatizado, esto no siempre fue así y aún existe mucha controversia en su uso ya que no gozan de las mismas garantías de aprendizaje que otros algoritmos. Algunos libros sobre fundamentos de aprendizaje automatizado incluso se abstienen de incluir un tratamineto sobre redes neuronales ya que la teoría del aprendizaje computacional [47] desarrollada en 1984 no ofrece fundamentos sólidos al respecto. Ésta es, en parte, la razón de que hoy en día se hable sobre el tema con escepticismo y cautela.

La historia sobre el desarrollo de las redes neuronales comienza en 1943 cuando McCulloch y Pitts desarrollan un modelo matemático que consistía de una suma de las entradas ponderada por un conjunto de *pesos* y cuya salida consistía de una función de activación simple [48]. Aunque el modelo ya había sido planteado,

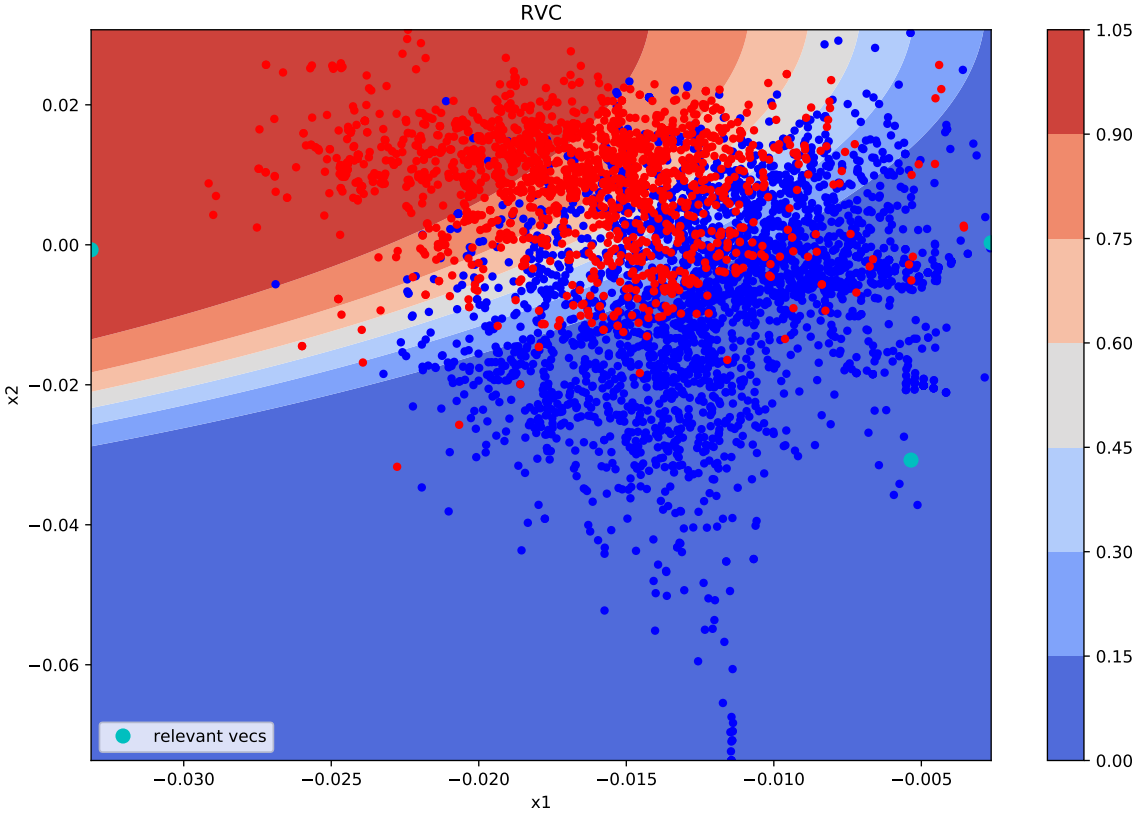


Figura 3.6: Solución definida por una máquina de relevancia vectorial con un kernel radial y 3 vectores de relevancia (derecha) para la base de datos SPAM.

aún no estaba claro cómo estimar los *pesos* que ponderaban las entradas del algoritmo. Fue hasta 1958, cuando Frank Rosenblatt consiguió estimar los *pesos* del modelo a través de un algoritmo que denominó *perceptrón* [49], que se obtuvo una aproximación concreta a lo que se conocería más tarde como "red neuronal". Sin embargo, el camino en el desarrollo de las redes neuronales no fue rápido y fluido. En 1969 Minsky y Papert publicaron un famoso libro llamado "Perceptrones", en el cual se argumentaba que el algoritmo desarrollado por Frank Rosenblatt presentaba fuertes limitaciones cuando el conjunto de datos no era linealmente separable [50]. Esto trajo como consecuencia el olvido y la falta de interés respecto al tema durante varios años, y no fue sino hasta 1986 cuando Rumelhart, Hinton y Williams llaman de nuevo la atención de la comunidad científica al inventar el algoritmo de retropropagación que permitía ajustar modelos con capas ocultas [51]. En este punto las redes neuronales habían llegado para quedarse con más fuerza y seguir evolucionando a la par de los sistemas computacionales y la tecnología.

Esta sección tiene como objetivo presentar el modelo de "red neuronal" o "perceptrón multicapa" y definir algunas características del modelo, así como varios detalles importantes sobre su implementación. Se comenzará por plantear el modelo más básico de red neuronal conocido como *perceptrón* y posteriormente se desarrollará su versión más general: el perceptrón multicapa.

3.5.1. Perceptrón

El perceptrón es uno de los primeros algoritmos desarrollados en el aprendizaje automatizado. Este modelo define una hipótesis de la forma $\hat{y}(x) = signo(w \cdot x)$ (i.e. la salida del algoritmo es 1 o -1), donde los pesos w son determinados usando un método de descenso por el gradiente estocástico [43]. El perceptrón busca los pesos en el modelo al minimizar la función de pérdida definida por:

$$F(w) = \frac{1}{T}\sum_{t=1}^{T} \text{máx}(0, -y_t(w \cdot x_t)) \tag{3.27}$$

Bajo esta notación, y_t representa la clase a la que pertenece la t-ésima observación en la muestra. El perceptrón recibe una observación x_t a la vez y mide su error $\tilde{F}(w,x_t) = \text{máx}(0, -y_t(w \cdot x_t))$ en cada iteración. Si el algoritmo clasifica la observación x_t correctamente entonces $\tilde{F}(w,x_t) = 0$ y los pesos no son actualizados, de lo contrario $\tilde{F}(w,x_t) = -y_t(w \cdot x_t)$ y $\nabla\tilde{F}(w,x_t) = -y_t \cdot x_t$. En este caso los pesos se actualizan de acuerdo a la fórmula $w_{t+1} = w_t - \eta\nabla\tilde{F}(w,x_t) = w_t + \eta y_t \cdot x_t$ donde $\eta > 0$ es un parámetro libre conocido como tasa de aprendizaje. Cuando la muestra de entrenamiento es linealmente separable el algoritmo converge en un número finito de pasos, en otro caso se establece un criterio de paro después de haberse cumplido algún número de iteraciones.

Es interesante notar que, cuando algún punto x_t es clasificado incorrectamente, la cantidad $y_t(w_t \cdot x_t)$ será negativa, y el algoritmo actualizará esta cantidad en la siguiente iteración al añadir un término positivo, a saber $\eta \cdot \|x_t\|^2 > 0$:

$$\begin{aligned} y_t(w_{t+1} \cdot x_t) &= y_t(w_t \cdot x_t) + \eta y_t^2 \cdot \|x_t\|^2 \\ &= y_t(w_t \cdot x_t) + \eta \cdot \|x_t\|^2 \end{aligned} \tag{3.28}$$

Esta especie de "corrección", aunque resulta muy intuitiva, no implica de ninguna manera que la cantidad $y_t(w_{t+1} \cdot x_t)$ será positiva, solamente que ha sido parcialmente corregida al añadir a ésta un término positivo. Generalmente los valores iniciales de los pesos son fijados en cero, aunque pueden elegirse de manera aleatoria de acuerdo a una distribución Gaussiana de media cero.

3.5.2. Perceptrón multicapa

El perceptrón multicapa, también conocido como red neuronal feed-forward, no es más que una función no lineal de un conjunto de variables de entrada $\{x_i\}$ al de salida $\{y_i\}$ controlada por un conjunto de parámetros $\{w_i\}$ conocidos como pesos. En este caso no se tiene convexidad y en consecuencia será probable obtener mínimos locales de la función de error. El modelo básico de red neuronal comprende cuatro etapas importantes que se describen a continuación:

1) Sean $x_1, ..., x_D$ las características de entrada. Para cada $j = 1, ..., M$, se construyen las activaciones a_j como combinaciones lineales de estas características:

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

(Esta es la primera capa de la red)

2) Para $j = 1, ..., M$, se aplica una función de activación no lineal h y se definen las unidades ocultas z_j:

$$z_j = h(a_j)$$

3) Para $k = 1, ..., K$, donde K es el número de salidas de la red, se generan las nuevas unidades de activación a_k como combinaciones lineales de las unidades ocultas z_j:

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

(Esta es la segunda capa de la red)

4) Finalmente las unidades de activación son transformadas usando una función de activación apropiada que mapee al conjunto de salida:

$$y_k = \sigma(a_k)$$

para $k = 1, ..., K$

Así, el modelo toma la forma:

$$y_k(x,w) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right)$$

La red se evalúa usando los cuatro pasos explicados anteriormente y en general este proceso es conocido como propagación hacia adelante. Si se define $x_0 = 1$, el modelo toma la forma más simplificada:

$$y_k(x,w) = \sigma\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right)$$

Este es, por mucho, el modelo de red neuronal más comúnmente aplicado en la práctica, aunque se puede generalizar al añadir más capas ocultas.

3.5.2.1. Medidas de error y funciones de activación

Es importante establecer cuál será la medida de error que se buscará minimizar en el modelo. Las medidas del error para los diferentes casos (clasificación binaria, clasificación multiclase o regresión) surgen de manera natural al considerar la función de verosimilitud para cada caso. En el caso de regresión se asumirá, como casi siempre, una distribución Gaussiana con media en $y(x_n,w)$ y parametrizada con una precisión β:

$$\begin{aligned} P(t\,|x,w,\beta) &= \prod_{n=1}^{N} P(t_n\,|x_n,w,\beta) \\ &= \prod_{n=1}^{N} N\left(t_n\,\middle|\,y(x_n,w),\beta^{-1}\right) \\ &= \left(2\pi\beta^{-1}\right)^{-\frac{N}{2}} \exp\left(\frac{-\beta}{2}\sum_{n=1}^{N}\left(t_n - y(x_n,w)\right)\right) \end{aligned} \tag{3.29}$$

El error que se buscará minimizar estará dado por la log-verosimilitud negativa:

$$-\log(P) = \frac{N}{2}\log\left(2\pi\beta^{-1}\right) + \frac{\beta}{2}\sum_{n=1}^{N}\left(t_n - y(x_n,w)\right)^2$$

Esto implica que maximizar la verosimilitud es equivalente a minimizar la suma de los cuadrdados del error dada por:

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\left(t_n - y(x_n,w)\right)^2 \tag{3.30}$$

Es importante notar que como $y(x_n,w)$ no es lineal, entonces $E(w)$ no será convexo y en la práctica se encontrará por lo general un mínimo local. Para el caso de la regresión la función de activación de la salida será la identidad, es decir $y_k = a_k$. Además, dada la ecuación (3.30), se tiene:

$$\frac{dE(w)}{da_k} = y_k - t_k \tag{3.31}$$

Este hecho será aprovechado posteriormente en la derivación del error de propagación. Se considerará ahora el caso de la clasificación binaria. En este caso la variable respuesta es dicotómica y la función de verosimilitud está dada por un modelo Bernoulli:

$$P(t\,|x,w,\beta) = \prod_{n=1}^{N} P(t_n\,|x_n,w,\beta) = \prod_{n=1}^{N} y(x_n,w)^{t_n}\cdot\left(1 - y(x_n,w)\right)^{1-t_n}$$

Se sigue que en este caso la función del error $E(w)$ está dada por la entropía cruzada:

$$-\log(P) = -\sum_{n=1}^{N} [t_n \cdot \log(y(x_n, w)) + (1 - t_n) \cdot \log(1 - y(x_n, w))] = E(w) \tag{3.32}$$

La función de activación de salida usual en este caso es la sigmoide logística:

$$y = \sigma(a) \equiv \frac{1}{1 + \exp(-a)} \tag{3.33}$$

Y la interpetación de la salida $y(x_n, w)$ será en términos de la probabilidad condicional de pertenecer a la clase 1 dado el vector de entradas: $P(C_1 | x)$. En este caso también se cumple la propiedad (3.31). En efecto, derivando la función de activación de salida (3.33) se tiene :

$$\begin{aligned} \frac{dy_k}{da_k} = \frac{\exp(-a_k)}{(1 + \exp(-a_k))^2} &= \left(1 - \frac{1}{1 + \exp(-a_k)}\right)\left(\frac{1}{1 + \exp(-a_k)}\right) \\ &= (1 - y_k) y_k \end{aligned} \tag{3.34}$$

Se sigue de la ecuación del error de entropía cruzada (3.32) que:

$$\begin{aligned} \frac{dE(w)}{da_k} &= -\left[\frac{t_k}{y_k} \cdot \frac{\delta y_k}{\delta a_k} + \frac{1 - t_k}{1 - y_k} \cdot -\frac{\delta y_k}{\delta a_k}\right] \\ &= \left[\frac{1 - t_k}{1 - y_k} - \frac{t_k}{y_k}\right] \frac{\delta y_k}{\delta a_k} \\ &= \left[\frac{(1 - t_k) y_k - t_k (1 - y_k)}{(1 - y_k) y_k}\right] (1 - y_k) y_k \\ &= y_k - t_k y_k - t_k + t_k y_k \\ &= y_k - t_k \end{aligned} \tag{3.35}$$

Observe que para simplificar la notación en los cálculos anteriores se ha definido $y_k = y(x_k, w)$ y no debe confundirse esta notación con la del caso de la clasificación multiclase en el que se tienen K variables de salida y_k.

Las propiedades de la clasificación multiclase se derivan de manera similiar a los casos anteriores. Bajo este contexto se generan las variables dummy $t_k \in \{0, 1\}$ y $y(x_n, w)$ se interpreta como $P(t_k = 1 | x)$. La medida del error nuevamente se deriva de la menos log-verosimilitud y se usa la función softmax como función de activación de la salida. El cuadro 3.1 resume los resultados obtenidos hasta este punto.

Medidas de error y funciones de activación		
Problema	Función de activación $\sigma(a)$	Función de error $E(w)$
Clasificación binaria	$\frac{1}{1+\exp(-a)}$	$-\sum_{n=1}^{N} \log\left(y(x_n, w)^{t_n} \cdot (1 - y(x_n, w))^{1-t_n}\right)$
Clasificación multiclase	$\frac{\exp(a_k(x,w))}{\sum_j \exp(a_j(x,w))}$	$-\sum_{n=1}^{N} \sum_{k=1}^{K} t_{kn} \log(y_k(x_n, w))$
Regresión	a	$\frac{1}{2} \sum_{n=1}^{N} (t_n - y(x_n, w))^2$

Tabla 3.1: Resumen de las medidas del error y funciones de activación de salida para diferentes problemas de redes neuronales [52].

Una vez que se han definido las medidas de error apropiadas según el contexto, el objetivo se centra ahora en encontrar el vector de pesos w tal que minimize $E(w)$. Al no tener la propiedad de convexidad el problema de optimización se vuelve complicado. Casi todos los métodos proponen un enfoque numérico en el que los

pesos se actualizan iterativamente de acuerdo a la siguiente fórmula recursiva:

$$w^{(i+1)} = w^{(i)} + \Delta w^{(i)}$$

Dado que el gradiente puede proporcionar la dirección de máximo descenso en una función, los métodos más populares proponen aprovechar la información del gradiente de la función del error para calcular $\Delta w^{(i)}$ en la fórmula anterior. Usualmente se usa un método conocido como optimización de descenso iterando la fórmula:

$$w^{(i+1)} = w^{(i)} + \eta \nabla E\left(w^{(i)}\right)$$

Esta fórmula calcula el error $\nabla E(w)$ usando toda la muestra de entrenamineto a la vez. Una estrategia de aprendizaje en línea ha mostrado dar mejores resultados. Este enfoque actualiza los pesos calculando el gradiente del error $\nabla E_n(w)$ para el n-ésimo punto de la muestra de entrenamineto en cada iteración hasta recorrer la muestra completa. La fórmula bajo este contexto estará indexada también por el elemento de la muestra que se esté considerando en la i-ésima iteración:

$$w^{(i+1)} = w^{(i)} + \eta \nabla E_n\left(w^{(i)}\right) \tag{3.36}$$

Aunque otras variantes más eficientes de este algoritmo han sido perfeccionadas con el paso del tiempo, en este trabajo se adoptará esta aproximación ya que es una de las más sencillas y ha mostrado dar buenos resultados en la práctica en cuestión de precisión y eficiencia computacional. Los puntos de la muestra, al considerar la fórmula (3.36), pueden ser seleccionados secuencialmente o bajo un esquema de muestreo aleatorio simple con reemplazo.

3.5.2.2. Error de retropropagación

La sección anterior tuvo como objetivo definir las bases matemáticas y algorítmicas sobre las que opera el modelo. Sin embargo una pregunta aún ha quedado sin responder: ¿Cómo calcular el gradiente de la función del error $\nabla E_n(w)$ de la fórmula (3.36)?. En efecto, esta es la única pieza que falta para completar el modelo. Esta sección tiene como objetivo presentar el método conocido como "Error de retropopagación" en respuesta a esta pregunta.

Comenzaremos por suponer que la función de error asociada puede expresarse como una suma de los errores en cada punto de la muestra:

$$E(w) = \sum_{n=1}^{N} E_n(w)$$

Evidentemente todas las funciones de error definidas en la sección anterior pertenecen a esta familia tan amplia de funciones. El objetivo es entonces evaluar el gradiente del error asociado a cada punto en la muestra.

Comenzaremos por suponer que la propagación hacia adelante ha sido efectuada en la red neuronal, i.e. las unidades ocultas y las funciones de activación han sido todas calculadas a partir de las fórmulas:

$$a_j = \sum_i w_{ji} z_i \tag{3.37}$$

$$z_j = h(a_j) \tag{3.38}$$

En la primera iteración del algoritmo la propagación hacia adelante es efectuada a partir de las fórmulas (3.37) y (3.38) inicializando los pesos w con valores cercanos a cero. Una elección común consiste en generar valores aleatorios de una distribución normal de media cero y varianza pequeña.

Como una consecuencia sencilla de la regla de la cadena tenemos:

$$\frac{dE_n}{dw_{ji}} = \frac{dE_n}{da_j}\frac{da_j}{dw_{ji}}$$

Sea $d_j = \frac{dE_n}{da_j}$ y observe que según la ecuación (3.37) $\frac{da_j}{dw_{ji}} = z_i$. Entonces:

$$\frac{dE_n}{dw_{ji}} = d_j z_i \tag{3.39}$$

Los valores z_i son conocidos porque ya han sido evaluados en el proceso de propagación hacia adelante. Así, lo único que hace falta conocer de la ecuación (3.39) son los valores $d_j = \frac{dE_n}{da_j}$. De la ecuación (3.31) se tiene $d_k = y_k - t_k$. Por lo tanto, lo único que resta es evaluar d_j para cada una de las capas ocultas.

Vamos a suponer el caso más general en el que la red tiene K salidas, por ejemplo en el problema de clasificación multiclase se tienen K probabilidades de salida, una por cada clase, y en este caso el error E_n asociado a cada elemento de la muestra es la suma de los errores en cada salida (tal como se muestra en la tabla 3.1). Tomando en cuenta lo anterior, y dado que cada unidad de salida dependerá a su vez de cada unidad oculta, se sigue de la regla de la cadena que los valores d_j para las unidades ocultas son de la forma:

$$d_j = \sum_k \frac{dE_n}{da_k}\frac{da_k}{da_j} \tag{3.40}$$

Observe que según las ecuaciones (3.37) y (3.38) se tiene $a_k = \sum_i w_{ki} h(a_i)$. De la ecuación (3.40) obtenemos entonces la fórmula de retropropagación:

$$d_j = \sum_k d_k \frac{da_k}{da_j} = \sum_k d_k w_{kj} h'(a_j) = h'(a_j) \sum_k d_k w_{kj}$$

Una vez obtenidos todos los valores d_j, estos son sustituidos en la ecuación (3.39) para obtener las derivadas parciales $\frac{dE_n}{dw_{ji}}$. Estas derivadas parciales son usadas finalmente para construir el gradiente $\nabla E_n(w)$ que podrá ser usado en el algoritmo de optimización de descenso por el gradiente a partir de la fórmula recursiva (3.36).

3.5.3. Detalles de implementación

Una implementación exitosa de un modelo de red neuronal no se sigue necesariamente de un conjunto de reglas establecidas, en muchos casos incluso la experiencia y la práctica con este tipo de modelos serán los únicos que determinarán el éxito del modelo. Existen, sin embargo, una serie de recomendaciones básicas que pueden servir como guía en la mayoría de las situaciones.

- *Pesos iniciales:* Si los pesos son cercanos a cero, el modelo de red neuronal será aproximadamente lineal. Así, conforme los pesos se incrementan, el modelo se vuelve no-lineal sólo donde sea necesario. Comúnmente se eligen los pesos iniciales de manera aleatoria y cercanos a cero, esto se hace por lo general simulando de una distribución Gaussiana.

- *Sobre-ajuste (overfitting):* Los problemas de sobre-ajuste se pueden presentar en casi todos los algoritmos. En el caso de las redes neuronales son frecuentes cuando la solución se acerca al mínimo global. Se recomienda determinar el número de iteraciones necesarias por validación crcuzada. Algunos algoritmos añaden incluso un término de regularización al error para penalizar la complejidad del modelo, evitando así problemas de sobre-ajuste.

- *Escala de los datos:* Cuando las variables de entrada están medidas en diferentes unidades resulta mejor aplicar un tratamineto previo a los datos. Comúnmente se opta por estandarizar la base restando a cada entrada su media y dividiendo entre su desviación estándar.

- *Número de unidades y capas ocultas:* El Teorema de aproximación universal [53] establece que cualquier función continua puede ser aproximada con el grado de error deseado a partir de una red neuronal con una capa oculta y un número finito de unidades ocultas. Esto en general puede motivarnos a adoptar una postura conservadora y optar por usar una sola capa oculta en el modelo. En la práctica la elección de un número mayor de capas ocultas puede producir también resultados exitosos pero deberá estar guiada siempre por la experimentación y un conocimiento profundo del problema. Al fijar una capa oculta, el número de unidades ocultas será un parámetro que puede ser estimado usando validación cruzada.

- *No-convexidad:* A diferencia de otros algoritmos, las redes neuronales no garantizan unicidad en la solución obtenida. Las funciones de error son no convexas y por ello existe el peligro de quedar atrapado en un mínimo local. Comúnmente se opta por correr el algoritmo con diferentes valores iniciales y tomar la solución que presente un mejor rendimiento en la muestra de prueba.

3.6. AdaBoost

¿Puede un conjunto de clasificadores débiles crear un clasificador más exacto?, esta fue la pregunta formulada por Kearns a través de su artículo "*Thoughts on hypothesis Boosting*" [54] en 1988. El objetivo era "impulsar" una hipótesis con un rendimiento poco mejor que una decisión aleatoria, hacia una hipótesis capaz de cumplir en rigor con el concepto de aprendizaje probable aproximadamente correcto [55] desarrollado por Valiant en 1984. Dos años más tarde, en 1990, Robert Shapire respondería a la pregunta a través de su artículo "*The strength of weak learnability*" [56] en el cual establecía formalmente la equivalencia entre el aprendizaje débil y fuerte. El resultado de esta investigación fue uno de los algoritmos más exitosos y bien motivados dentro del aprendizaje automatizado: el "AdaBoost" o "Boosting adaptativo". Eso le valió a Shapire de un premio Gödel en 2003 y motivó el desarrollo de una amplia familia de algoritmos de aprendizaje conocida como *Boosting*. Esta sección se centra en presentar el algoritmo AdaBoost. Otras versiones populares del algoritmo como "LogitBoost" o "Gradient Boosting" pueden ser encontradas en [28].

3.6.1. Antecedentes teóricos

Esta sección tiene como objetivo introducir algunos conceptos y definiciones básicas referentes a la teoría del aprendizaje computacional [47]. En general estos conceptos son aplicables a un gran número de algoritmos de aprendizaje automatizado. Las máquinas de soporte vectorial, por ejemplo, se ven beneficiadas de muchas de estas propiedades teóricas aunque no son necesarias para entender su funcionamiento algorítmico. El desarrollo del AdaBoost estuvo directamente motivado por estos conceptos y por ello será necesario abordar, aunque muy brevemente, esta teoría.

En la teoría del aprendizaje computacional supervisado se asume naturalmente la existencia de un algoritmo A que es entrenado a partir de un **conjunto de ejemplos**, generalmente conocido como **muestra de entrenamiento**, con el fin de hacer predicciones sobre nuevas entradas. Las salidas del algoritmo son conocidas como **hipótesis** y se asume que éstas han sido seleccionadas de un conjunto de hipótesis H. Se supone también la existencia de una función c que determina la manera en que cada entrada es asociada a una salida, ésta es la función que el algoritmo deberá "aprender" y es conocida como **concepto objetivo**. El concepto objetivo pertenece a su vez a una clase de conceptos C. Es natural pensar que nuestro algoritmo no siempre acertará en sus predicciones y por ello es necesario considerar la probabilidad de cometer un error. La siguiente definición pone en claro este concepto.

Definición 3.1 (Error de generalización) *Sea $h \in H$ la hipótesis seleccionada por el algoritmo A y $c \in C$ el concepto objetivo. Si D es una distribución sobre x, se define el error de generalización del algoritmo A respecto de la hipótesis h como:*

$$R(h) = \underset{x\sim D}{P}\left(h(x) \neq c(x)\right) = \underset{x\sim D}{E}\left[\mathrm{I}_{\{h(x)\neq c(x)\}}\right]$$

En general no será posible calcular el error de generalización de manera explícita ya que ello requiere conocer la distribución D. En su lugar tendremos que conformarnos con una aproximación insesgada de éste.

Definición 3.2 (Error empírico) *Sea $h \in H$ la hipótesis seleccionada por el algoritmo A y $c \in C$ el concepto objetivo. Si $S = \{x_1, \ldots, x_n\}$ es un conjunto de elementos de entrada, se define el error empírico del algoritmo A respecto de la hipótesis h y el conjunto S como:*

$$\hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{I}_{\{h(x_i)\neq c(x_i)\}}$$

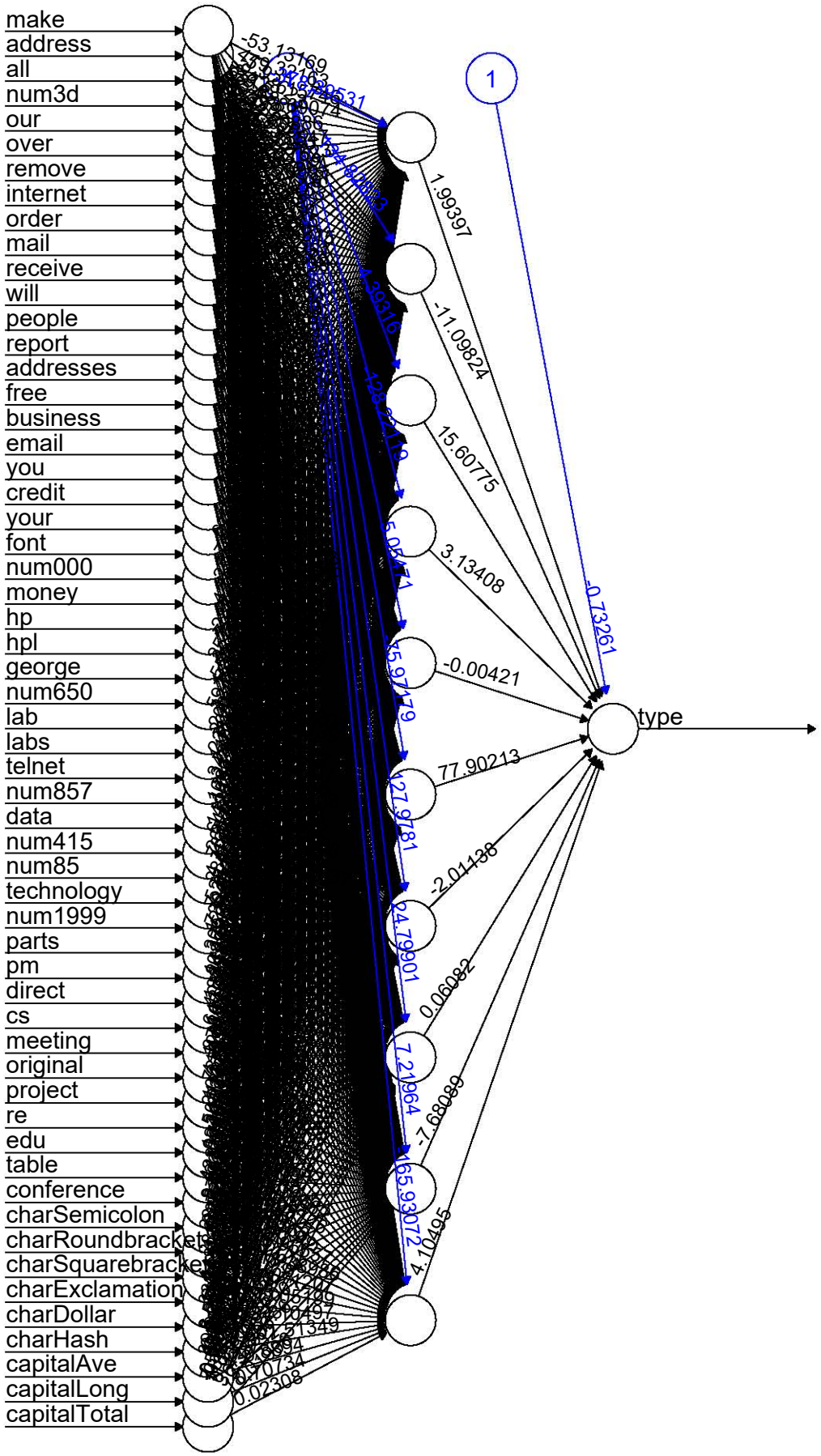


Figura 3.7: Representación gráfica de una red neuronal entrenada para resolver el problema de la clasificación de spam. Las entradas de la red son las frecuencias de las palabras en los correos. La arquitectura de la red consta de una capa oculta con 10 unidades ocultas y la salida es el tipo de correo (“spam” o “no spam”). La función de activación seleccionada es la sigmoide logística y se ha usado la entropía cruzada como medida del error.

Es sencillo probar que el error empírico es un estimador insesgado del error de generalización, i.e. $E\left[\hat{R}(h)\right] = R(h)$. Las definiciones anteriores permiten reconocer el error cometido por el algoritmo. Sin embargo, es deseable esperar que entre mejor se entrene al algoritmo menor será dicho error. En otras palabras, desearíamos poder disminuir el error cometido tanto como quisieramos al aumentar la muestra de entrenamiento. La siguiente definición formaliza estas nociones en el concepto de *aprendizaje probable aproximadamente correcto*.

Definición 3.3 (Aprendizaje PAC) *Una clase de conceptos C se dice que cumple con aprendizaje probable aproximadamente correcto si existe un algoritmo A y una función polinomial poli*$(\cdot,\cdot,\cdot,\cdot)$ *tal que* $\forall \varepsilon, \delta > 0$*, y para cualquier distribución D en una muestra S y cualquier concepto objetivo* $c \in C$*, se cumple para cualquier tamaño de muestra m tal que* $m \geq poli(1/\varepsilon, 1/\delta, n, size(c))$*:*

$$\underset{S\sim D^m}{P}\left[R(h_S) \leq \varepsilon\right] \geq 1-\delta$$

La definición anterior significa que con una probabilidad de al menos $1-\delta$ el algoritmo produce una hipótesis que es "aproximadamente correcta" al considerar que comete un error de a lo más ε. La función $size(c)$ corresponde al tamaño de la representación más pequeña del concepto c en el esquema de representación que se haya adoptado, por ejemplo en un esquema de representación binaria la función $size(c)$ corresponderá a la longitud medida en bits.

A pesar de su simpleza, la definición anterior resulta, por lo general, muy restrictiva para la mayoría de los algoritmos de aprendizaje. Existe una noción de aprendizaje menos restrictiva conocida como "aprendizaje débil" y es por ello que el concepto anterior también es conocido como "Aprendizaje fuerte PAC".

Definición 3.4 (Aprendizaje débil PAC) *Una clase de conceptos C se dice que cumple con aprendizaje débil probable aproximadamente correcto si existe un algoritmo A, un real* $\gamma > 0$ *y una función polinomial poli*$(\cdot,\cdot,\cdot)$ *tal que* $\forall \delta > 0$*, y para cualquier distribución D en una muestra S y cualquier concepto objetivo* $c \in C$*, se cumple para cualquier tamaño de muestra m tal que* $m \geq poli(1/\delta, n, size(c))$*:*

$$\underset{S\sim D^m}{P}\left[R(h_S) \leq \frac{1}{2} - \gamma\right] \geq 1-\delta$$

Las hipótesis definidas por un algoritmo de aprendizaje débil son llamadas clasificadores base. Esta noción, aunque en apariencia es menor restrictiva, es de hecho equivalente al aprendizaje fuerte. Intuitivamente un algoritmo que cumple con aprendizaje débil es aquel que tiene un rendimiento poco mejor que el simple hecho de tomar una decisión aleatoria.

3.6.2. Construcción del algoritmo

Se asumirá que se tiene un conjunto de clasificadores base. La idea es combinar los clasificadores base para obtener una predicción más precisa. Así, la hipótesis generada por el AdaBoost será una ponderación de cada uno de los clasificadores base. Formalmente, sean $h_t\colon t = 1,...,T$ los clasificadores base y $\alpha_t\colon t = 1,...,T$ los pesos asociados. La hipótesis será entonces de la forma:

$$g(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

El proceso funciona de forma iterativa generando una hipótesis a la vez. Al final de cada iteración o "ronda de Boosting" se da más peso a aquellos ejemplos en la muestra que han sido clasificados erroneamente por el algoritmo. Así, en la ronda siguiente los clasificadores base se enfocarán más en aquellos puntos que han sido mal clasificados en rondas pasadas. La distribución de los pesos se actualiza en cada ronda de Boosting para $i = 1,...,n$ de acuerdo a la pérdida exponencial de la siguiente manera:

$$D_{t+1}(i) \leftarrow \frac{D_t(i)\exp\left(-\alpha_t\left(y_i \cdot h_t(x_i)\right)\right)}{Z_t} \tag{3.41}$$

Los valores Z_t simplemente son factores de normalización que permiten asegurar que la nueva distribución también sume uno. Si definimos $\varepsilon_t = \underset{i\sim D_t}{P}(h_t(x_i) \neq y_i)$, i.e. ε_t es el error de generalización cometido en la t-ésima iteración asumiendo la distribución D_t, entonces los factores de normalización Z_t pueden ser determinados como:

$$\begin{aligned} Z_t &= \sum_{i=1}^{n} D_t(i)\, e^{-\alpha_t y_i h_t(x_i)} \\ &= \sum_{i:y_i h_t(x_i)=+1} D_t(i)\, e^{-\alpha_t} + \sum_{i:y_i h_t(x_i)=-1} D_t(i)\, e^{\alpha_t} \\ &= (1-\varepsilon_t)\, e^{-\alpha_t} + \varepsilon_t e^{\alpha_t} \end{aligned} \tag{3.42}$$

Para determinar el valor de los pesos α_t observe primero que a partir de una aplicación iterativa de la fórmula (3.41) se obtiene lo siguiente:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)\exp\left(-\alpha_t\left(y_i \cdot h_t(x_i)\right)\right)}{Z_t} \\ &= \frac{D_{t-1}(i)\exp\left(-\alpha_{t-1}\left(y_i \cdot h_{t-1}(x_i)\right)\right)}{Z_{t-1}} \cdot \frac{\exp\left(-\alpha_t\left(y_i \cdot h_t(x_i)\right)\right)}{Z_t} \\ &= \frac{D_{t-1}(i)\exp\left(-y_i\left(\alpha_t \cdot h_t(x_i) + \alpha_{t-1} \cdot h_{t-1}(x_i)\right)\right)}{Z_t \cdot Z_{t-1}} \\ &\vdots \\ &= \frac{\exp\left(-y_i \sum_{j=1}^{t} \alpha_j \cdot h_j(x_i)\right)}{n\prod_{j=1}^{t} Z_j} \\ &= \frac{e^{-y_i \cdot g_t(x_i)}}{n\prod_{j=1}^{t} Z_j} \end{aligned} \tag{3.43}$$

Observe que el factor n^{-1} se obtiene de la distribución inicial, la cual se ha supuesto uniforme (i.e. $D_1 = \frac{1}{n}$). De la ecuación (3.43) se obtiene:

$$e^{-y_i \cdot g_t(x_i)} = nD_{t+1}(i)\prod_{j=1}^{t} Z_j \tag{3.44}$$

De la ecuación (3.44) y del hecho de que $\mathrm{I}_{\{u\leq 0\}} \leq e^{-u}\ \forall u$, es posible derivar una cota para el error empírico de la siguiente manera:

$$\begin{aligned} \hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{I}_{\{h(x_i)\neq c(x_i)\}} &= \frac{1}{n}\sum_{i=1}^{n} \mathrm{I}_{\{y_i g(x_i)\leq 0\}} \\ &\leq \frac{1}{n}\sum_{i=1}^{n} e^{-y_i \cdot g_t(x_i)} \\ &= \frac{1}{n}\sum_{i=1}^{n}\left[nD_{t+1}(i)\prod_{j=1}^{t} Z_j\right] \\ &= \prod_{j=1}^{t} Z_j\left[\sum_{i=1}^{n} D_{t+1}(i)\right] \\ &= \prod_{j=1}^{t} Z_j \end{aligned} \tag{3.45}$$

$$\therefore \hat{R}(h) \leq \prod_{j=1}^{t} Z_j \tag{3.46}$$

Por lo tanto es razonable elegir los pesos α_t de tal manera que se minimize la cota del error empírico (3.46). Esto es equivalente a minimizar Z_t. Derivando parcialmente tenemos:

$$\frac{\delta Z_t}{\delta \alpha_t} = -(1-\varepsilon_t)e^{-\alpha_t} + \varepsilon_t e^{\alpha_t} \tag{3.47}$$

Igualando a cero la ecuación (3.47) y resolviendo para α_t:

$$\begin{aligned}
&\varepsilon_t e^{\alpha_t} = (1-\varepsilon_t)e^{-\alpha_t} \\
&\log(\varepsilon_t) + \alpha_t = \log(1-\varepsilon_t) - \alpha_t \\
&2\alpha_t = \log\left(\tfrac{1-\varepsilon_t}{\varepsilon_t}\right) \\
&\alpha_t = \tfrac{1}{2}\log\left(\tfrac{1-\varepsilon_t}{\varepsilon_t}\right)
\end{aligned}$$

Se concluye entonces que los pesos del modelo están dados por un medio del logaritmo del momio. Estos pesos pueden ser sustituidos en la ecuación (3.42) para reevaluar los factores de normalización:

$$\begin{aligned}
Z_t &= (1-\varepsilon_t)e^{-\alpha_t} + \varepsilon_t e^{\alpha_t} \\
&= (1-\varepsilon_t)\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)^{-\frac{1}{2}} + \varepsilon_t\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)^{\frac{1}{2}} \\
&= ((1-\varepsilon_t)\varepsilon_t)^{\frac{1}{2}} + ((1-\varepsilon_t)\varepsilon_t)^{\frac{1}{2}} \\
&= 2\sqrt{(1-\varepsilon_t)\varepsilon_t}
\end{aligned} \tag{3.48}$$

Los resultados anteriores pueden ser interpretados como una manera de deducir el algoritmo secuencialmente. Se procede ahora a resumir el funcionamiento algorítmico del AdaBoost con base en los resultados obtenidos.

- Suponga que se ha recibido una muestra de entrenamiento $S = \{(x_1, y_1), ..., (x_n, y_n)\}$.

1) Para $i = 1, ..., n$ se define la distribución inicial uniforme:

$$D_1(i) \leftarrow \frac{1}{n}$$

2) Para la iteración $t = 1, ..., T$ se genera el clasificador base h_t, su peso asociado α_t, el factor de normalización Z_t y se actualiza la distribución D_{t+1} que se usará en la siguiente iteración de acuerdo a las fórmulas:

$$\begin{aligned}
&h_t \leftarrow \text{ clasificador base que minimiza el error } \varepsilon_t = \underset{i\sim D_t}{P}(h_t(x_i) \neq y_i) \\
&\alpha_t \leftarrow \tfrac{1}{2}\log\left(\tfrac{1-\varepsilon_t}{\varepsilon_t}\right) \\
&Z_t \leftarrow 2\sqrt{(1-\varepsilon_t)\varepsilon_t}
\end{aligned}$$

   * Para $i = 1, ..., n$ actualizar los pesos en la distribución de acuerdo a la pérdida exponencial y los factores de normalización Z_t:

$$D_{t+1}(i) \leftarrow \frac{D_t(i)\exp(-\alpha_t(y_i \cdot h_t(x_i)))}{Z_t}$$

3 Devolver como resultado la hipótesis h generada por el algoritmo a partir de los pesos α_t y las hipótesis h_t:

$$h \leftarrow signo(g(x)) = signo\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

3.6.3. Detalles de implementación

Aunque el AdaBoost se ve beneficiado de una rica teoría y diversas garantías de aprendizaje, hay algunos detalles de implementación que deben ser considerados cuando se busca aplicar exitosamente el algoritmo a problemas reales. El modelo deja libre la elección del clasificador base y en general no está claro cómo construir uno. Aún cuando se pudiera elegir un clasificador base lo más probable es que éste tuviera a su vez un conjunto de parámetros libres que necesitarían ser determinados de alguna manera. Éste y otros problemas típicos de implementación son descritos brevemente a continuación.

- *Clasificadores base:* Los algoritmos de aprendizaje débil que comúnmente se eligen en la implementación del AdaBoost forman parte de una amplia familia conocida como árboles de decisión. Los árboles de decisión son algoritmos de aprendizaje supervisado que definen particiones jerárquicas del espacio de entrada. Lo más común es usar un tipo particular de árbol de decisión conocido como tope de decisión. Un tope de decisión no es más que un árbol de decisión de un solo nivel que clasifica una observación considerando una sola covariable. Aunque es posible usar árboles de mayor profundidad en problemas simples, se debe considerar el costo computacional de implementarlos en problemas de alta dimensionalidad. En estos casos árboles con complejidades mayores pueden resultar ineficientes y por ello se opta por seleccionar un conjunto de árboles de poca profundidad con un número suficientemente grande de rondas de boosting.

- *Rondas de Boosting:* El número de rondas de boosting es un parámetro libre en el modelo que debe ser seleccionado de manera cuidadosa. Un número insuficiente de rondas puede provocar un desempeño deficiente del algoritmo mientras que un número excesivo de ellas puede conducir a problemas de sobreajuste. Por lo general el número de rondas de boosting se determina a partir de un criterio de paro que consiste en identificar el momento en que el error de validación cruzada comience a incrementar.

- *Ruido:* Se ha probado de manera empírica que la presencia de ruido entre las clases puede disminuir severamente la exactitud del algoritmo. Una solución puede ser elegir una función objetivo “menos agresiva” que la pérdida exponencial o añadir un término de regularización para penalizar pesos muy grandes. Exposiciones más generales del algoritmo que permiten intercambiar la función del error y añadir términos de regularización pueden ser encontradas en [57].

3.7. Nota bibliográfica

Por simplicidad se ha adoptado la notación de [2] en el desarrollo de la sección 3.1. El experimento realizado en esta misma sección está basado en los resultados presentados en [58]. Una excelente exposición teórica de las máquinas de soporte vectorial puede ser encontrada en [43], la sección 3.3 está basada en esta fuente. La discusión sobre las máquinas de relevancia vectorial fue tomada del trabajo original de Michael E. Tipping en [59] y de la explicación detallada de Tristán Fletcher en [60]. En [61] puede encontrarse un capítulo dedicado esencialmente a la presentación teórica de las máquinas de relevancia vectorial. Una exposición clara sobre el clasificador Naive Bayes puede ser encontrada en [3] y la teoría aquí presentada está basada en esta fuente. La teoría estadística de las redes neuronales presentada en este capítulo está basada en [52]. Un tratamiento más extenso sobre redes neuronales puede ser encontrado en [62]. En particular la exposición teórica del perceptrón fue tomada de [43]. El fundamento teórico del algoritmo AdaBoost aquí presentado estuvo totalmente motivado por el planteamiento sobre este tema expuesto en [43].

Capítulo 4

Técnicas para evaluación de modelos

La evaluación del modelo es una de las etapas más importantes en el diseño e implementación de un algoritmo de aprendizaje. Es en esta etapa en la cual se podrá medir el rendimiento del algoritmo y sus capacidades de predicción. Además, los parámetros libres en el modelo podrán ser ajustados de tal manera que se alcance un mejor desempeño. Este breve capítulo tiene como objetivo detallar algunos de los métodos de evaluación de modelos más importantes, mismos que serán usados en las aplicaciones de este trabajo.

4.1. Conjuntos de entrenamiento, validación y prueba

Bajo el esquema clásico del aprendizaje supervisado, se parte de una muestra de ejemplos. Este conjunto de ejemplos serán "aprendidos" por el algoritmo con el fin de seleccionar una hipótesis capaz de hacer predicciones sobre elementos no observados. Así, se busca una hipótesis que generalice un patrón en los datos. Una hipótesis que memoriza en vez de generalizar, aprenderá también el ruido en los datos y dará un mal desempeño cuando elementos no observados le sean presentados. Con el fin de medir las capacidades de generalización del algoritmo ante elementos no observados se divide la muestra en dos partes. La primera de éstas será usada para entrenar al algoritmo y la segunda para probar su desempeño. Así, se tiene una muestra de entrenamiento y una de prueba. Es importante resaltar que la muestra de prueba no deberá ser nunca utilizada en la fase de entrenamiento, ya que esto no permitiría medir de manera justa las capacidades de generalización del algoritmo.

Además de los parámetros propios del modelo, se tendrán por lo general ciertos *parámetros libres*. Por ejemplo, en una máquina de soporte vectorial se tendrá un parámetro de costo C asociado a las variables de holgura en la función objetivo del problema de optimización convexa y, si además se ha decidido usar algún kernel, se tendrán otros paramétros adicionales. Estos parámetros libres no son estimados directamente por el modelo y por ello tendrán que ser seleccionados cuidadosamente bajo algún criterio. Idealmente los mejores parámetros serán aquellos que maximicen las capacidades de generalización del modelo. Lo más común es fijar un conjunto de candidatos y seleccionar sólo aquella combinación de parámetros que reporte el mejor rendimiento en un conjunto de datos previamente clasificados, este proceso recibe el nombre de selección de parámetros. Evidentemente no podemos usar la muestra de prueba en el proceso de selección de parámetros ya que estaríamos contaminando la fase de entrenamiento con información de la fase de prueba. Es necesario entonces subparticionar la muestra de entrenamiento con el fin de generar otra muestra en la que podamos probar el desempeño del algoritmo bajo ciertos parámetros. Esta nueva submuestra recibe el nombre de muestra de validación y será usada exclusivamente para el proceso de selección de parámetros.

Por lo general una mayor cantidad de datos será destinada a la fase de entrenamiento y selección de parámetros, ya que de ello dependerá directamente el desempeño del algoritmo. La mayoría de los autores recomiendan destinar un 50 % a la muestra de entrenamiento, un 25 % a la muestra de validación y un 25 % a la muestra de prueba. Cuando se cuenta con una cantidad limitada de datos estas proporciones pueden cambiar de tal manera que una mayor parte de la muestra sea destinada al entrenamiento del algoritmo.

4.2. Validación cruzada

Dado que en muchos escenarios se cuenta con una cantidad limitada de datos, se han desarrollado técnicas de remuestreo que permiten fusionar las fases de entrenamiento y selección de parámetros en una sola. Una de las técnicas más usadas consiste en particionar la muestra de entrenamiento en k subconjuntos independientes. Luego, uno de estos subconjuntos es seleccionado como una muestra de validación y el algoritmo es entrenado en el resto de ellos. El proceso es repetido para cada uno de los k subconjuntos de la muestra y una medida del error es aplicada en cada uno de ellos. Al final se promedia el error obtenido en cada subconjunto para obtener un error global, el cual recibe el nombre de error de validación cruzada. Este proceso es aplicado a cada una de las posibles combinaciones de los parámetros propuestos. Finalmente se selecciona aquella combinación de parámetros que reporten el error de validación cruzada más pequeño. Este proceso es conocido como validación cruzada.

También es común calcular la varianza del error para cada conjunto de parámetros y reportarla junto con el error promedio. El modelo es luego entrenado en la muestra completa sin particionar y con los parámetros seleccionados, para luego evaluar su desempeño final en la muestra de prueba.

4.3. La matriz de confusión

Una manera de reportar los resultados obtenidos en la muestra de prueba es mediante una matriz que registre los elementos que fueron correctamente e incorrectamente clasificados. Existen dos tipos de errores que el algoritmo podría cometer durante el proceso de clasificación. Si C_1 y C_2 son las posibles clases en las que un elemento puede ser clasificado, entonces el algoritmo podría asignarle a este elemento la clase C_1 cuando en realidad pertenece a la clase C_2 y viceversa. Si C_1 es la clase de referencia, entonces los elementos mal clasificados bajo estos escenarios reciben el nombre de falsos positivos (FP) y falsos negativos (FN) respectivamente. Naturalmente los elementos clasificados correctamente son verdaderos positivos (VP) y verdaderos negativos (VN) según sea el caso. Estos valores pueden ser acomodados en un arreglo matricial de dos columnas y dos renglones, en el que los renglones representan los valores de referencia y las columnas los predichos por el algoritmo. Este arreglo recibe el nombre de matriz de confusión. La figura 4.1 ilustra esta situación cuando la clase de referencia (o clase positiva) es C_1.

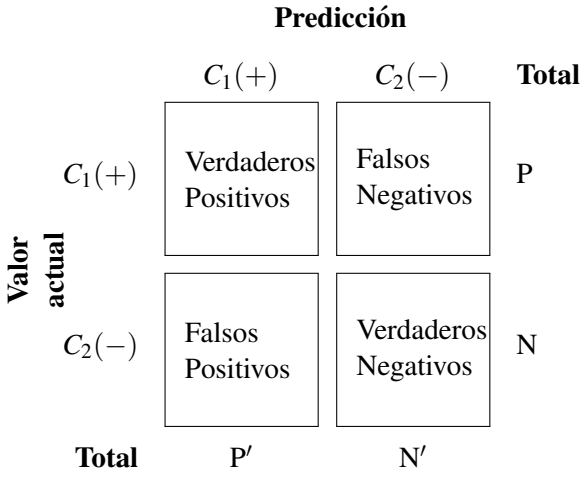


Tabla 4.1: Matriz de Confusión

La matriz de confusión de un algoritmo que nunca se equivoca tendría solo valores en la diagonal. Por lo general se espera que un algoritmo con un rendimiento considerable registre valores pequeños fuera de la diagonal. La figura 4.2 muestra un caso típico de este comportamineto bajo el contexto de la detección de spam. En este caso la clase positiva o de referencia es “spam”. Para extender esta representación al problema multiclase basta con agregar tantos renglones y columnas como clases se tengan.

		Predicción		
		spam	**no spam**	**Total**
Valor actual	**spam**	157	11	168
	no spam	30	1195	1225
	Total	187	1206	

Tabla 4.2: Matriz de Confusión SPAM

4.4. Métricas de exactitud y precisión

Aunque la matriz de confusión ofrece una representación clara de los resultados en el conjunto de prueba, es deseable resumir la información contenida en ella a un sólo valor que especifique el desempeño global del algoritmo. Es por ello que una variedad extensa de métricas han sido diseñadas para medir el rendimiento del algoritmo bajo diferentes contextos. De la matriz de la tabla 4.1 es posible por ejemplo calcular una tasa global de mala clasificación o una tasa de falsos positivos o falsos negativos. Algunas de las métricas más usuales son definidas a continuación:

$$\begin{aligned} \text{Exactitud} &= \frac{VP+VN}{VP+FN+FP+VN} \\ \text{Sensibilidad} &= \frac{VP}{VP+FN} \\ \text{Especificidad} &= \frac{VN}{FP+VN} \\ \text{Precisión} &= \frac{VP}{VP+FP} \end{aligned}$$

La exactitud registra la tasa global de mala clasificación, es decir divide el número de elementos clasificados correctamente entre el total de elementos. Aunque puede parecer intuitiva, la exactitud no es del todo adecuada bajo escenarios más específicos, pues resume la información de cuatro números en uno solo. Es por ello que en muchos casos se prefiere fijar la atención a métricas que resuman sólo parcialmente la matriz de confusión. La sensibilidad por ejemplo considera sólo el primer renglón de la matriz, y puede ser interpretada como la tasa de verdaderos positivos. Análogamente la especificidad calcula la tasa de verdaderos negativos. La precisión, por su lado, indica la proporción de aciertos del algoritmo respecto del total de elementos que ha clasificado como positivos.

La precisión y la sensibilidad guardan una estrecha relación, ya que si el número de falsos negativos crece entonces el número de falsos positivos decrece y viceversa. Por ello es común resumir la información contenida en estas métricas a una sola. Así, la métrica F_1 puede ser definida como la media armónica de la sensibilidad y la especificidad de la siguiente manera:

$$F_1 = 2\frac{\text{Precisión} \cdot \text{Sensibilidad}}{\text{Precisión + Sensibilidad}}$$

Cuando la muestra no está balanceada las medidas anteriores pueden no resultar del todo adecuadas. En este caso se recomienda usar la exactitud balanceda como medida del rendimiento del algoritmo.

$$\text{Exactitud Balanceada} = \frac{\text{Sensibilidad} + \text{Especificidad}}{2}$$

Note que cuando la muestra está perfectamente balanceada, la exactitud balanceada es igual a la exactitud. En muestras no balanceadas se recomienda medir la precisión por clase, individual y promedio.

4.5. Métricas sensibles al costo

En algunos contextos los costos de mala clasificación no son iguales. En el problema de la detección de spam, por ejemplo, resulta más grave clasificar un correo legítimo como spam que dejar que algunos correos no deseados pasen el filtro, ya que esto provocaría que algún correo importante no fuera recibido en el momento adecuado o incluso nunca ser leído. Es por ello que resulta importante considerar algunas métricas que asignen diferentes costos de mala clasificación de acuerdo al problema del que se trate. Dado $\lambda > 0$, definimos la exactitud ponderada de la siguiente manera:

$$\text{Exactitud Pond}\,(\lambda) = \frac{VP + \lambda \cdot VN}{VP + FN + \lambda \cdot (FP + VN)}$$

La exactitud ponderada trata a cada elemento de la clase negativa como si fueran λ elementos, por lo que cuando un falso positivo ocurre, este es contado como si fueran λ errores; y cuando es clasificado correctamente este cuenta como λ aciertos. Así, en el caso de la detección de spam, cada correo mal clasificado como spam (falsos positivos) tendrá un peso λ veces mayor. Dada la exactitud ponderada, es posible también definir su recíproco:

$$\begin{aligned} \text{Error Pond}\,(\lambda) &= 1 - \text{Exactitud Pond}\,(\lambda) \\ &= \frac{VP + FN + \lambda \cdot (FP + VN) - VP - \lambda \cdot VN}{VP + FN + \lambda \cdot (FP + VN)} \\ &= \frac{FN + \lambda \cdot FP}{VP + FN + \lambda \cdot (FP + VN)} \end{aligned} \tag{4.1}$$

Así, equivalentemente al maximizar la exactitud ponderada se busca minimizar el error ponderado. Con el fin de definir una métrica que sea de utilidad para medir la efectividad de un filtro de spam podemos considerar el caso base para el error ponderado. En este contexto el caso base es aquel filtro de spam que no hace absolutamente nada, i.e. deja pasar todos los correos clasificándolos como "no spam". Entonces, el filtro del caso base no clasifica ningún correo legítimo como spam ($FP = 0$) y por lo tanto se tiene:

$$\text{ErrorPond}(\lambda)_{Base} = \frac{FN}{VP + FN + \lambda \cdot (FP + VN)} = \frac{\#\text{SPAM}}{VP + FN + \lambda \cdot (FP + VN)}$$

Donde "#SPAM" representa el número total de correos clasificados como "spam" en la muestra de entrenamiento. Tomando como referencia el caso base anterior, se define la tasa de costo total como el cociente:

$$TCT_\lambda = \frac{\text{ErrorPond}(\lambda)_{Base}}{\text{Error Pond}\,(\lambda)} = \frac{\#\text{SPAM}}{FN + \lambda \cdot FP}$$

La tasa de costo total es ampliamente usada para medir la efectividad de los filtros de spam. Aunque la exactitud ponderada también puede ser usada, es fácil malinterpretarla bajo este contexto en particular. Es importante notar que $TCT_\lambda > 1$ cuando el filtro ofrece un rendimiento mejor que el caso base, es decir que es mejor implementar el filtro a no implementarlo y por ello un buen filtro debería tener una tasa de costo total mucho mayor que 1 para poder ser considerado últil en una aplicación real.

4.6. Curva ROC y área bajo la curva

La curva característica operativa del receptor o curva ROC (por su acrónimo en inglés *Receiver Operating Characteristic Curve*) muestra la sensibilidad del clasificador al graficar la tasa de falsos positivos *vs* la tasa de

verdaderos positivos. Así, la curva ROC muestra cuántas clasificaciones positivas correctas pueden ser ganadas al permitir más falsos positivos. Idealmente se espera que la curva ROC se encuentre cercana a uno. Cuando la curva se encuentra sobre la diagonal el rendimiento del algoritmo es igual al de un lanzamiento aleatorio. Con el fin de obtener un solo número que resuma el comportamiento de la curva ROC es común calcular su área. Así, valores cercanos a uno del área bajo la curva ROC son indicadores de un buen desempeño del clasificador, mientras valores cercanos a 0.5 indican una mala capacidad de clasificación. Además, el área bajo la curva ROC puede ser interpretada como la probabilidad de que un clasificador asigne un valor a alguna observación positiva seleccionada de manera aleatoria más alto que a una observación negativa.

A manera de ejemplo considere el problema de la detección de spam. Para ello se ha entrenado una máquina de soporte vectorial con un kernel radial. Así, el algoritmo asignará a los mensajes un valor numérico a partir del cual se podrá determinar la clase a la que pertenece. Por ejemplo si este valor es mayor que cero, entonces el correo será clasificado como spam. En otro caso pasará el filtro y será enviado a la bandeja principal como un mensaje legítimo. El valor a partir del cual el clasificador toma una decisión es conocido como punto de corte y en el ejemplo anterior es igual a cero. En un caso ideal, se esperaría que el algoritmo asigne valores positivos únicamente a aquellos correos que en efecto son spam y valores negativos en caso contrario. Sin embargo, por lo general el algoritmo cometerá cierto nivel de error y asignará valores negativos a correos que eran spam y valores positivos a correos que no lo eran. De esta manera, si se piensa que los valores que genera el algoritmo siguen una cierta distribución entonces se esperaría que esta fuera bimodal, i.e. se esperaría poder distinguir dos subpoblaciones en ella (a saber “spam” y no “spam”). La figura 4.1 muestra el histograma de las salidas generadas por una máquina de soporte vectorial con un kernel radial en la base de datos SPAM, nótese que la distribución en efecto es bimodal. Dado que las clases de los correos se conocen a priori, se han distinguido ámbos tipos de correo por color. Es posible notar que el algoritmo ha asignado valores positivos a la mayoría de los correos que en efecto eran spam y valores negativos a la mayoría de los que no lo eran, sin embargo una pequeña proporción de los correos han sido malclasificados (esto está representado gráficamente por las regiones de las distribuciones que se sobreponen). El punto de corte del algoritmo se muestra con una línea vertical que intersecta al eje horizontal en cero.

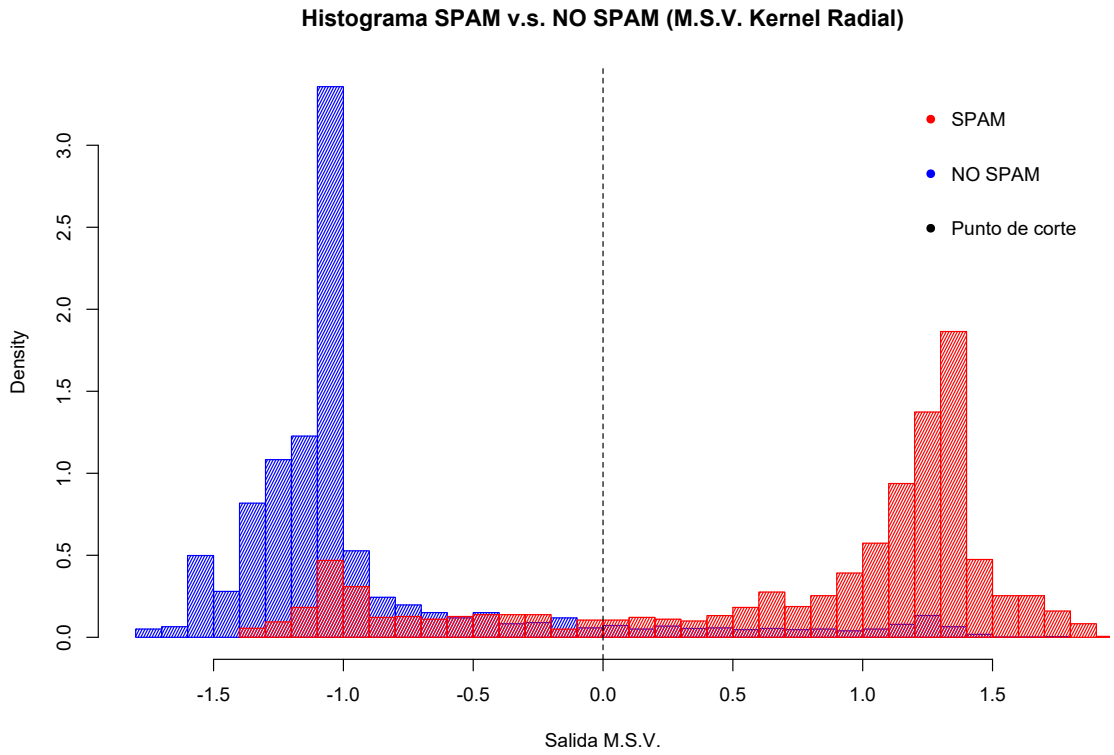


Figura 4.1: Histograma de las salidas generadas por una máquina de soporte vectorial con un kernel radial en la base de datos SPAM.

La curva ROC refleja el comportamiento del histograma en la figura 4.1 en el siguiente sentido. Considere las tasas de falsos positivos (TFP) y verdaderos positivos (TVP) definidas a continuación. Sean P y N el número total de observaciones positivas y negativas respectivamente en la muestra, entonces:

$$\begin{aligned} TFP &= \frac{FP}{N} \\ TVP &= \frac{TP}{P} \end{aligned} \tag{4.2}$$

Si el punto de corte del algoritmo está representado por la recta vertical que corta al eje horizontal en cero, tal como se muestra en la figura 4.1, entonces es posible calcular las tasas de falsos y verdaderos positivos a partir de las ecuaciones 4.2. En realidad no hay ninguna razón en particular para tomar al cero como punto de corte ya que esto dependerá del contexto del problema y el tipo de error que se prefiera minimizar. Así, desplazando el punto de corte de derecha a izquierda se obtiene un conjunto de valores para TFP y TVP. Es decir, para cada punto de corte $i \in \Re$ se tendrá una pareja de valores $(TFP, TVP)_i$. Graficando los puntos de la forma $(TFP, TVP)_i$ para cada i se obtiene la curva ROC (figura 4.2). Aunque es posible comparar el rendimiento de dos algoritmos a partir de sus curvas ROC, se prefiere calcular el área bajo la misma y compararlos en términos de esta área. Una curva ROC con valores cercanos a uno corresponde a una distribución bimodal en las salidas del algoritmo y por lo tanto a un buen clasificador. De esta manera, valores del área bajo la curva ROC cercanos a uno son indicadores de un mejor modelo.

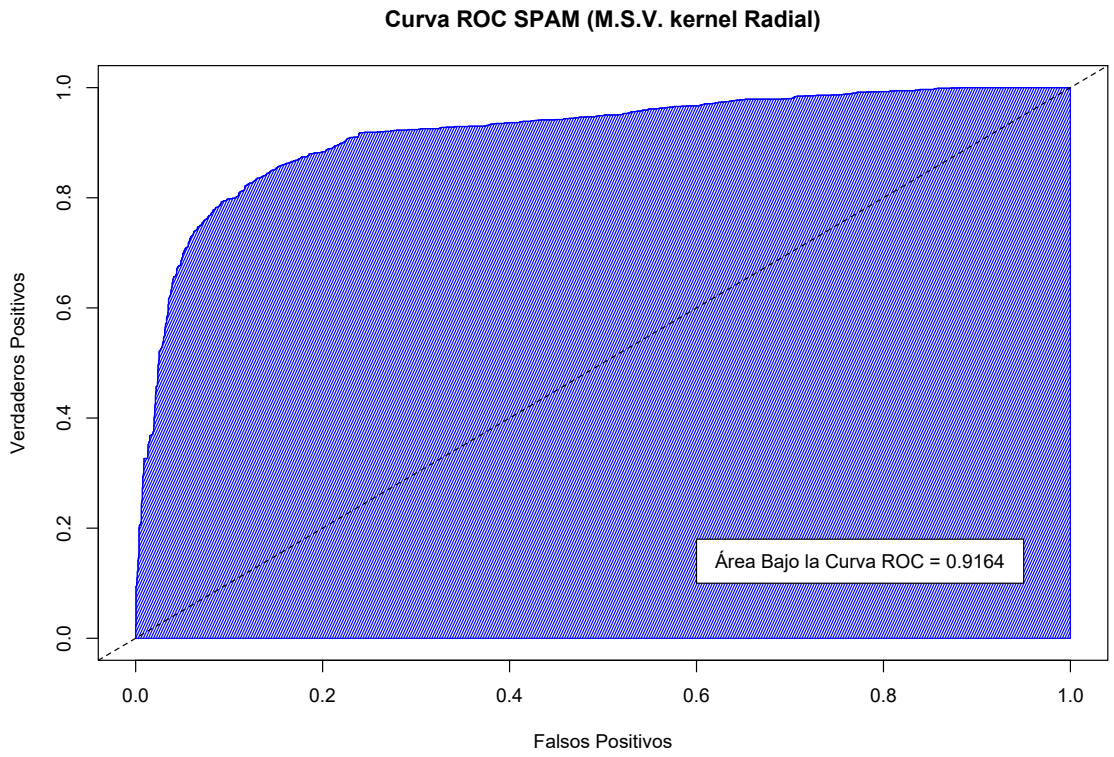


Figura 4.2: Curva ROC de una máquina de soporte vectorial con un kernel radial en la base de datos SPAM.

4.7. Nota bibliográfica

La mayoría de los conceptos aquí presentados son típicos en una introducción al aprendizaje automatizado y la minería de datos y pueden ser encontrados en casi cualquier fuente que trate sobre estos temas con mayor o menor profundidad. En general los conceptos básicos sobre valuación de modelos y técnicas de remuestreo pueden ser encontrados en [3]. Una exposición bien estructurada sobre estos temas está disponible en [63] y los conceptos referentes a métricas de exactitud y precisión han sido tomados de esta misma fuente. Un excelente

recurso que cubre de manera específica la mayoría de los tópicos fundamentales en técnicas de evaluación de modelos puede ser encontrado en [64]. Una exposición rápida y clara sobre validación cruzada es discutida en [43]. Las técnicas especiales de evaluación de modelos para filtros de spam fueron tomadas de [65]. Una explicación detallada sobre curvas ROC puede ser encontrada en [66] y una animación interactiva está disponible en [67].

Capítulo 5

Detección de SPAM

"Con el rápido crecimiento de Internet y los avances en la tecnología informática, el correo electrónico se ha convertido en una de las formas preferidas de comunicación e intercambio de información para fines comerciales y personales al ser rápido y conveniente. En años recientes, sin embargo, la eficacia y la confianza en el correo electrónico se han reducido muy notablemente por mensajes no deseados y no solicitados. Ahora, también se ha convertido en un medio primario para propagar estafas de publicidad y virus maliciosos." [23]

"El volumen mundial de SPAM ha aumentado considerablemente y durante el primer trimestre de 2008, el correo electrónico no deseado representó *más de nueve de cada diez* mensajes de correo enviados a través de Internet" [23]. Un caso reciente de spam fue protagonizado por el ransomware [1] *Wanna Cry*. Este es un tipo de virus del formato troyano con la capacidad de introducirse en nuestro equipo explotando una vulnerabilidad de software. El esquema de funcionamiento del ataque de *Wanna Cry* consistía en una serie de spam masivo a direcciones de correo electrónico con un enlace de descarga del dropper [2]. Cuando se descarga el archivo adjunto (dropper) se infecta el equipo con el virus *Wanna Cry*.

Aunque inicialmente el SPAM se presentaba de manera casi exclusiva en los correos electrónicos de los servidores, esto cambió con la llegada de los dispositivos móviles y el incremento en el número de páginas web. Ahora puede ser de interés filtrar los mensajes SMS recibidos por un teléfono inteligente o las publicaciones falsas distribuidas por algún "bot". Estos nuevos problemas pueden además darse en muy diversos escenarios, que van desde la política hasta el marketing.

Este capítulo tiene como objetivo ejemplificar el uso de los algoritmos de aprendizaje expuestos en capítulos anteriores bajo el contexto de la minería de textos a través de un problema clásico e interesante como el de la detección de SPAM. En específico, se entrenará un algoritmo que podrá ser usado para filtrar mensajes no deseados enviados a un teléfono móvil. Por simplicidad, la metodología usada en la selección y diseño del algoritmo ha sido dividida en varias secciones que se presentan a lo largo de este capítulo. La última sección representa un experimento independiente que tiene como objetivo comparar el rendimiento de una máquina de soporte vectorial y su versión Bayesiana, una máquina de relevancia vectorial.

Los capítulos anexos brindan información acerca del software y las bibliotecas usadas durante el desarrollo de este trabajo. Las técnicas de reducción de la dimensión basadas en modelación de tópicos latentes serán aprovechadas en la última sección de este trabajo. En particular se usó la descomposición espectral inducida por el análisis semántico latente con el fin de obtener representaciones gráficas de los correos y las fronteras de decisión definidas por los algoritmos, sin embargo, esta nueva representación no fue considerada para la comparación de los algoritmos ya que como tal no constituye un método de clasificación sino más bien un filtro de variables que puede ser usado de manera rápida.

[1] Un *ransomware* es un tipo de programa informático malintencionado que restringe el acceso a determinadas partes o archivos del sistema infectado, y pide un rescate a cambio de quitar esta restricción.

[2] Aplicación cuya función es instalar algún tipo de malware o virus en un sistema computacional objetivo.

5.1. Descripción de la base de datos

Con el fin de investigar el desempeño de los algoritmos de aprendizaje, se usarán dos bases de datos que pueden ser encontradas en [26] (*Machine Learning Repository*):

- *SMS SPAM Collection v.1*: Es un conjunto público de mensajes etiquetados con SMS que se han recopilado para la investigación de spam de teléfonos móviles. Cuenta con una colección compuesta por $5,574$ mensajes en inglés, mensajes reales y no clasificados, etiquetados según legítimo (ham) o spam. Este corpus ha sido recopilado de forma gratuita desde las siguientes fuentes de investigación en internet: 425 mensajes de spam SMS fueron extraídos manualmente del sitio web de Grumbletext; 3,375 mensajes de SMS seleccionados aleatoriamente del NUS SMS Corpus (que es un conjunto de datos de unos $10,000$ mensajes legítimos), 450 mensajes legítimos de SMS recolectados de la Tesis de Caroline Tag y finalmente, se ha incorporado el corpus SMS Spam v.0.1 Big, el cual tiene $1,002$ mensajes legítimos de SMS y 322 mensajes de spam. [68]

Atributo	Valor
Número de correos	4741
spam	728
no spam	4013
Número de atributos	122
¿Valores perdidos?	No
Fecha de donación	2012-06-22

Tabla 5.1: Descripción SMS Spam Collection v.1

- *SPAM E-mail Database*: Un conjunto de datos recogidos en Hewlett-Packard Labs, que clasifica 4601 correos electrónicos como spam o no-spam. Además de esta etiqueta de clase hay 57 variables que indican la frecuencia de ciertas palabras y caracteres en el e-mail. La colección de correos no deseados provino del administrador de correos y personas que habían reportado spam. La colección de correos electrónicos legítimos provino de trabajos archivados y correos electrónicos personales, por lo que la palabra 'george' y el código de área '650' son indicadores de no spam. Estos son útiles cuando se construye un filtro de spam personalizado. [69]

Atributo	Valor
Número de correos	4601
spam	1813
no spam	2788
Número de atributos	57
¿Valores perdidos?	No
Fecha de donación	1999-07-01

Tabla 5.2: Descripción Spam E-mail Database

La base de datos *SMS SPAM Collection v.1* será usada para entrenar un algoritmo de aprendizaje que funcione como un filtro de mensajes SMS recibidos por un teléfono móvil. En este caso los algoritmos candidatos serán un clasificador naive Bayes, una máquina de soporte vectorial, una red neuronal y el algoritmo AdaBoost. El experimento se desarrollará en las próximas secciones y tiene como objetivo comparar estos algoritmos en términos de la tasa de costo total definida en la sección 4.5. La base de datos *SPAM E-mail Database* será usada para realizar un sencillo experimento que tiene como objetivo comparar el desempeño de una máquina de soporte vectorial contra el de una máquina de relevancia vectorial. Este experimento será presentado en sección 5.4.5.

5.2. Preprocesamiento de la base de datos

La base de datos *SMS SPAM Collection v.1* se encuentra en su forma original como texto plano y por ello será necesario aplicar un tratamineto previo en este caso para que los algoritmos puedan operar en ella. La base de datos *SPAM E-mail Database* no necesita ningún tipo de preprocesamineto previo ya que ésta se ha encontrado en forma matricial. Tal como se ha descrito en la sección 2.1, una primera aproximación consiste en filtrar primero aquellas palabras que serán irrelevantes para el diseño del algoritmo. La tabla 5.3 muestra los primeros mensajes de la base de datos *SMS SPAM Collection v.1*.

Clase	Mensaje
no spam	Go until jurong point, crazy.. Available only in bugis n great world la e buffet...Cine [...]
no spam	Ok lar... Joking wif u oni...
spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to [...]
no spam	U dun say so early hor... U c already then say...
no spam	Nah I don't think he goes to usf, he lives around here though
spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some [...]

Tabla 5.3: Se muestran los primeros seis correos de la base de datos *SMS SPAM Collection v.1* en su forma no estructurada (texto plano)

En la mayoría de los casos el éxito de un filtro de SPAM depende de una selección cuidadosa de las variables que pueden aportar información valiosa al diseño del algoritmo. En este caso se observó que el uso de *emoticones*, caracteres especiales y abreviaciones comúnes se presentaban de manera típica en mensajes legítimos y por ello se optó primero por estandarizar esta información y cambiar cualquiera de los caracteres ":)", ":(", ":-)", ":-(" por la palabra "emoticon". De igual manera se estandarizó el uso de algunas abreviaciones. Así, términos como "2 you", "to u" ó "2 u" fueron cambiados por "to you".

Como se expuso en la sección 2.1, otras técnicas de preprocesamineto consisten en llevar a minúsculas los términos, quitar espacios en blanco, remover números y términos escasos, aplicar un método de lematización o usar un diccionario que permita remover palabras vacías. La tabla 5.4 resume las técnicas de preprocesamiento aplicadas a la base de datos.

Una vez que se han estandarizado los documentos según los procedimientos anteriores, se procede a adoptar la representación matricial expuesta en la sección 2.2. Como se mencionó anteriormente, esta matriz tiene diferentes características especiales como escasez, alta dimensionalidad y no negatividad. La figura 5.5 muestra la matriz de documento-término de la base de datos SMS SPAM Collection v.1 en la que se puede observar la presencia de las características antes mencionadas. El resultado final fue una matriz con un léxico de $1,286$ palabras.

Método	¿Se aplicó?
Remover números	SI
Remover puntuación	SI
Remover términos escasos	SI
n-gramas	NO
Lematización	SI
Remover espacios en blanco	SI
Cambiar a minúsculas	SI
Long. mín por término	SI (2)
Long. máx por término	NO

Tabla 5.4: Resumen del preprocesamiento aplicado a la base SMS SPAM Collection v.1

Matriz Documento-Término *SMS Spam Collection v.1*

type	alredy	also	always	amp	anything	around	ask	babe	...	week	well	will	win	won	work	yeah	yes
no spam	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
spam	0	0	0	0	0	0	0	0	...	0	0	0	1	0	0	0	0
no spam	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
no spam	0	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0
spam	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
no spam	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

Tabla 5.5: Se muestran los primeros seis renglones de la matriz *documento-término* de la base de datos *SMS SPAM Collection v.1* después de haber removido las palabras vacías y aplicado técnicas de preprocesamiento de texto como stemming o lematización. También se removieron los términos más escasos, la puntuación, los números y se cambiaron a minúsculas todos los caracteres.

5.3. Nubes de palabras

Dada la alta dimensionalidad del problema resulta útil observar algún gráfico que permita identificar patrones en la semántica de los datos. En este caso se ha optado por una nube de palabras (sección 2.4) discriminando por tipo de mensaje (“spam” o “no spam”). Se han elegido gamas de colores y posiciones distintas y los tamaños de las palabras están asociadas a sus frecuencias en la base de datos.

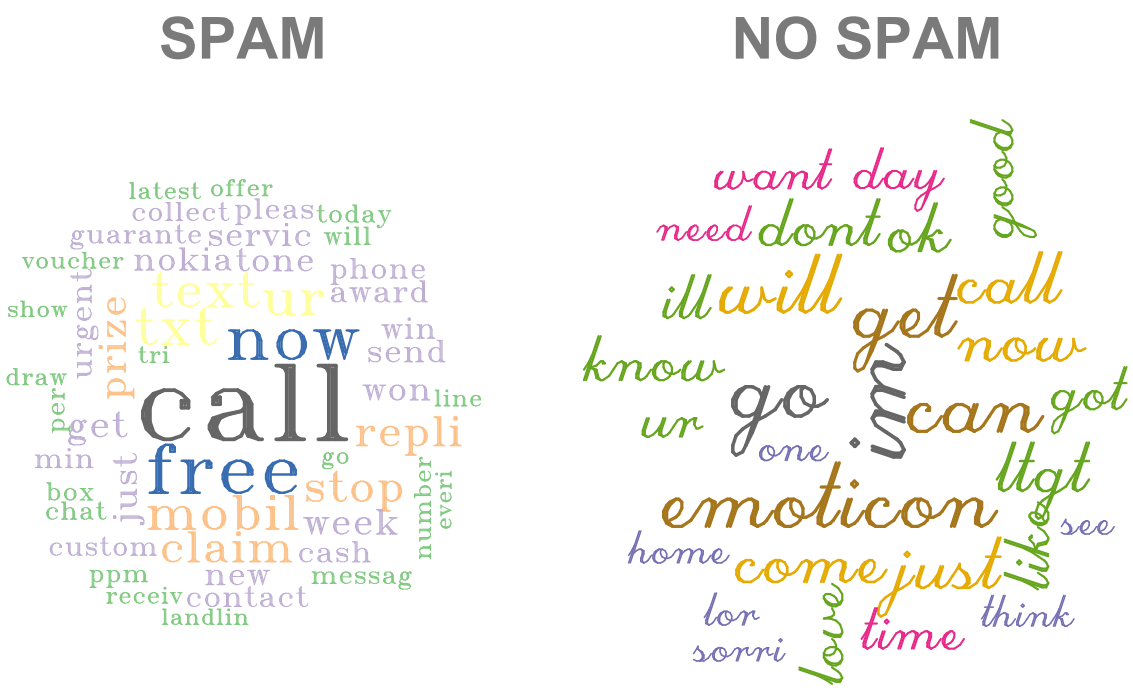


Figura 5.1: Nube de palabras de la base de datos *SMS Spam Collection v.1*. Se han construido dos nubes diferenciando por el tipo de correo: spam y no spam.

La figura 5.1 muestra que la semántica entre las dos clases de mensajes es diferente, por ejemplo los mensajes clasificados como spam tienden a mencionar palabras como “free”, “call” y “now” mientras que los mensajes legítimos usan un lenguaje más común que incluye palabras básicas como “love”, “will”, “get” y el uso frecuente de emoticones. Esto es un buen hallazgo, ya que en general el uso de una semántica distinta en los mensajes facilitará la tarea de clasificación para prácticamente cualquier algoritmo. Aunque otras características como la longitud del correo, el número de palabras mayúsculas o incluso el color o el tipo de fuente pueden ser añadidas para proporcionar información adicional al clasificador, esto no será aplicado en este caso porque mucha de esta información no está disponible en la base de datos original y porque las frecuencias por palabra son suficientes para discriminar bien entre ambas clases de correos (tal como sugieren los resultados obtenidos en la última sección).

5.4. Experimentos

5.4.1. ¿Qué metodo de selección de variables usar?

Los métodos de selección de variables expuestos en la sección 3.1 producen diferentes resultados de acuerdo al algoritmo seleccionado. Con el fin de evaluar el impacto de los métodos de selección de variables se entrenarán las versiones más simplificadas de los algoritmos considerando diferentes tamaños de léxicos que van desde las 50 hasta las 1250 palabras. El experimento se ha realizado considerando la tasa de costo total sin costos de mala clasificación ($\lambda = 1$) y con costos diferentes ($\lambda = 9$). Así, para cada uno de los cuatro algoritmos (Naive Bayes, MSV, redes neuronales y AdaBoost) se midió el desempeño a través de la tasa de costo total.

Al considerar la tasa de costo total con $\lambda = 1$ se puede notar que el algoritmo Naive Bayes presenta un mejor desempeño cuando se usa un léxico más grande (figura 5.2). Esto muestra que al considerar un número mayor de variables el algoritmo estima con más precisión las probabilidades de pertenencia a cada una de las clases. En este caso el peor desempeño lo reporta la medida de información mutua como técnica de selección de variables. El resto de las técnicas de selección muestran un comportamiento similar. La implementación de este algoritmo no tomó en cuenta ningún tipo de suavizamiento Laplaciano ya que al aplicarlo no se observó ninguna mejora significativa en el rendimiento.

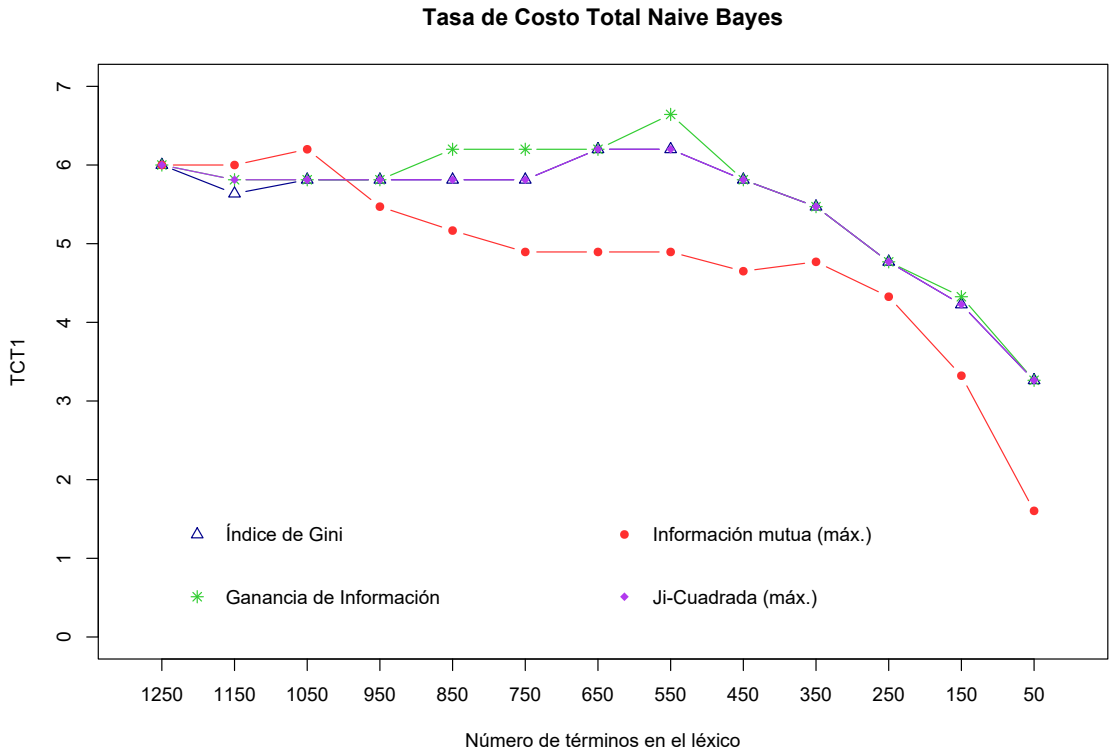


Figura 5.2: Tasa de Costo total con $\lambda = 1$ para el clasificador Naive Bayes sin suavizamiento Laplaciano usando varios métodos de selección de variables.

La máquina de soporte vectorial en combinación con la tasa de costo total con $\lambda = 1$ muestra que el algoritmo no se ve particularmente beneficiado por léxicos grandes (figura 5.3). Después de las 250 palabras el algoritmo no ofrece una mejora significativa que se vea reflejada en la tasa de costo total. Aunque en este caso no está del todo claro qué algoritmo de selección de variables puede ser mejor que otro, se aprecia una fuerte correlación entre los índices de Gini, Ji-cuadrada y ganancia de información. La información mutua reporta en muchos casos el peor rendimiento entre los cuatro métodos de selección. En este caso el algoritmo ha sido implementado en su versión más básica al considerar un kernel lineal tal como sugieren algunos autores, sin embargo la sección 5.4.2 explora otras posibilidades. El parámetro de costo *C* se fijó en uno para la construcción de la gráfica 5.3.

El escenario con las redes neuronales resulta ser más claro. Los mejores resultados se obtienen con los índices de Gini, ganancia de información y Ji-cuadrdada (figura 5.4). El peor rendimiento resulta ser el de la información mutua como método de selección de variables. Nuevamente la correlación de los tres mejores métodos es alta y en general no está claro cuál de ellos es el mejor pues presentan un desempeño muy similar. En este caso se ha entrenado un perceptrón como el modelo más simple de red neuronal y se ha usado el criterio de convergencia por defecto. Al igual que en el caso de las máquinas de soporte vectorial, el perceptrón tampoco parece verse beneficiado por léxicos grandes y siguiendo el principio de parsimonia optaríamos por léxicos no mayores a 750 palabras.

La tasa de costo total para el AdaBoost parece estabilizarse después de 450 palabras en el léxico (figura 5.5). La información mutua reporta el peor rendimiento en casi todos los casos. Se ha fijado en 30 el número de rondas de boosting y se han usado topes de decisión como clasificadores base.

El escenario cambia para el algoritmo Naive Bayes cuando los costos de mala clasificación se asumen distintos ($\lambda = 9$). En general se observa un mejor rendimiento con léxicos no mayores a 550 palabras (figura 5.6). Al igual que en el caso ($\lambda = 1$), la información mutua sigue sin mostrar un buen desempeño y el resto

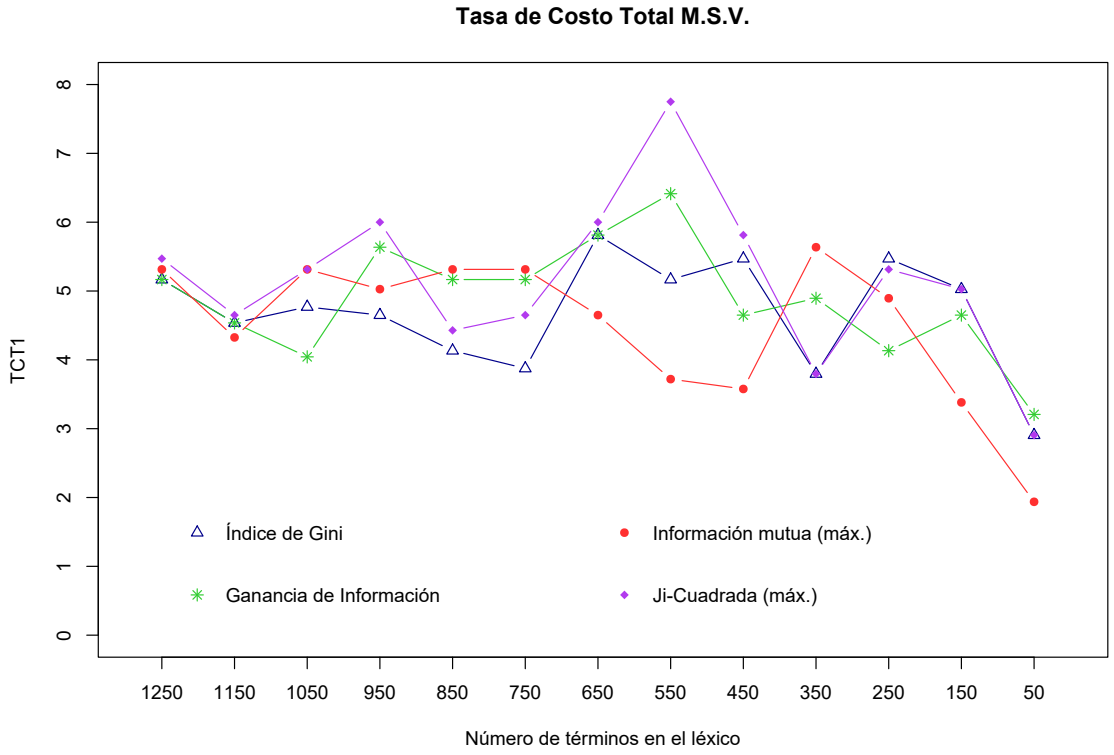


Figura 5.3: Tasa de Costo total con $\lambda = 1$ para una máquina de soporte vectorial con un kernel lineal usando varios métodos de selección de variables.

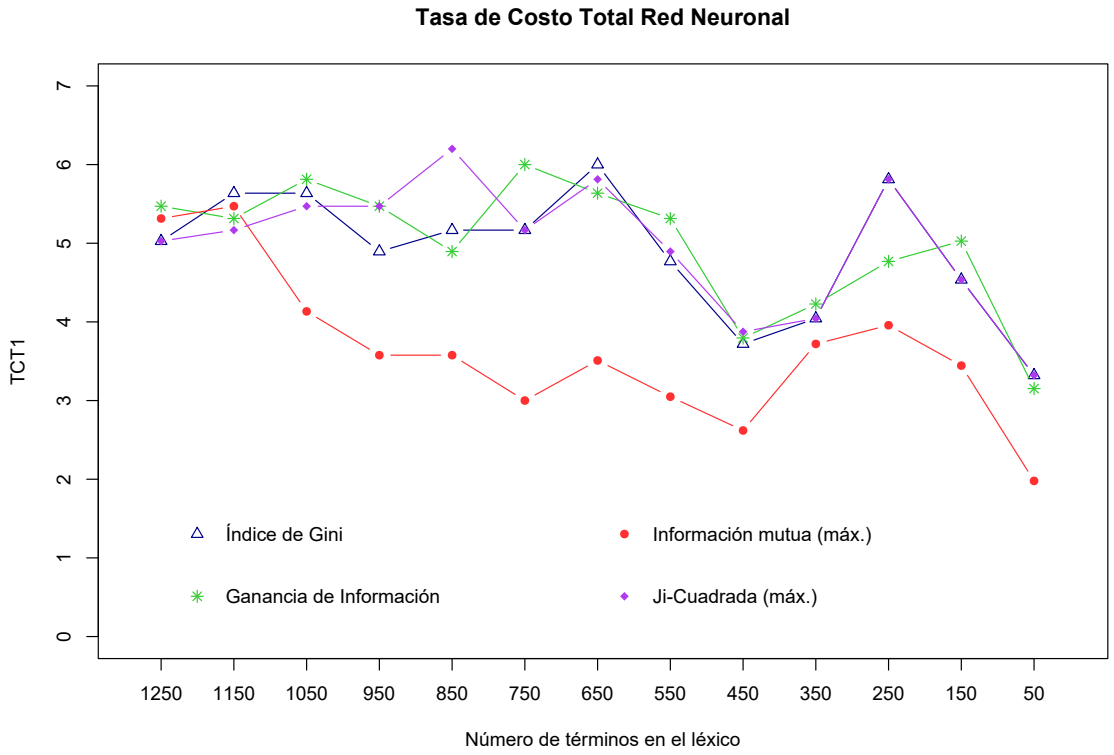


Figura 5.4: Tasa de Costo total con $\lambda = 1$ para una red neuronal sin ninguna capa oculta (perceptrón) usando varios métodos de selección de variables.

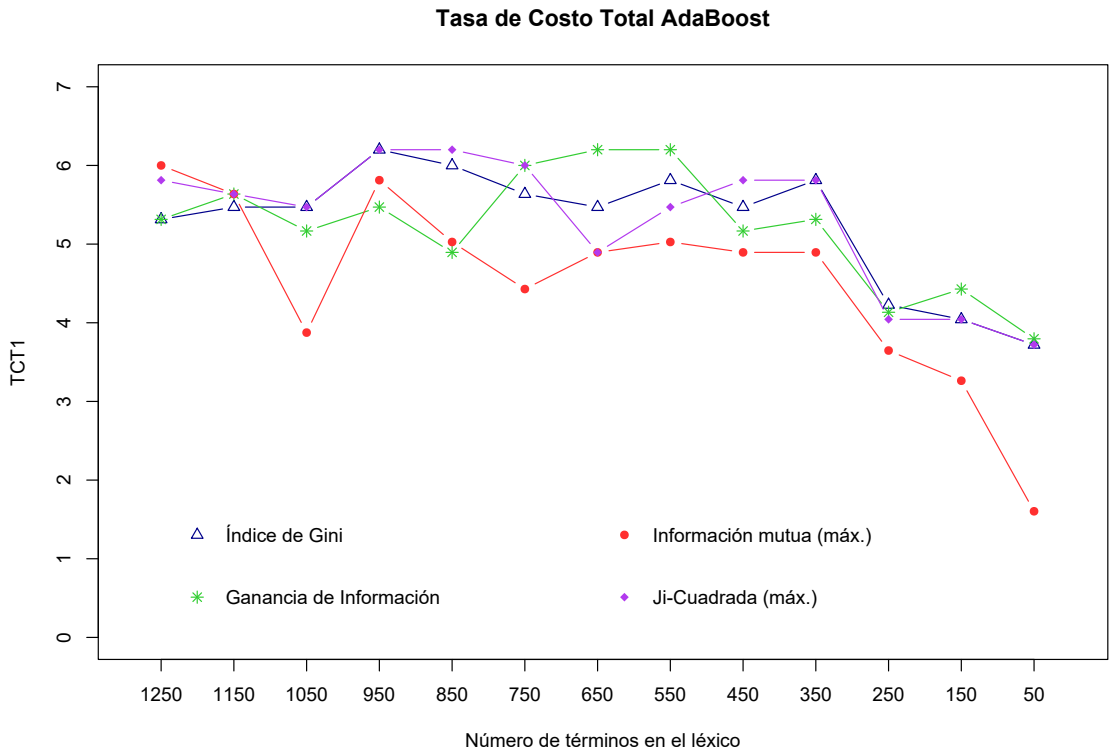


Figura 5.5: Tasa de Costo total con $\lambda = 1$ para el algoritmo AdaBoost con 30 rondas de boosting y topes de decisión como clasificadores base usando varios métodos de selección de variables.

de las medidas presentan una correlación alta. En general el algoritmo sigue teniendo un buen rendimiento ya que todas las métricas se encuentran por encima de la recta $y = 1$. Como se mencionó en la sección 4.5, esto significa que es mejor usar el filtro de SPAM a no usarlo.

La tasa de costo cotal con $\lambda = 9$ aplicada a una máquina de soporte vectorial con un kernel lineal no proporciona un buen resultado. Todos los algoritmos de selección de variables están sólo un poco por encima de la recta $y = 1$ (figura 5.7). Esto motiva a explorar otras opciones de funciones kernel y a ampliar la búsqueda de los parámetros en una muestra de validación con el fin de obtener resultados más competitivos. Esto se realizará de manera detallada en la sección 5.4.2.

La situación con la red neuronal es todavía peor. Con 350 y 450 palabras todos los algortimos de clasificación son inútiles con cualquiera de los métodos de selección de variables al considerar costos de mala clasificación desiguales (figura 5.8). Esta situación motiva la necesidad de explorar otras configuraciones en el modelo como la incorporación de alguna capa oculta y/o término de regularización en la función del error que penalice el ingreso de pesos demasiado grandes y evite así problemas de sobre-ajuste. Estas posibilidades se exploran en la sección 5.4.3.

El algoritmo AdaBoost muestra resultados aceptables al considerar costos de mala clasificación desiguales. A excepción de una ocasión, todos los algoritmos muestran buenos rendimientos con cualquiera de las técnicas de selección de variables al estar por encima de la recta $y = 1$. En este caso no está del todo claro cuál método de selección puede ser el mejor pues los resultados obtenidos en cada uno de ellos son muy similares y cualquiera de ellos podría funcionar bien para este algoritmo. El mejor rendimiento lo alcanza la ganancia de información con 650 variables.

Los resultados mostrados en esta sección tuvieron como objetivo comparar los distintos métodos de selección de variables. A excepción de la información mutua, el resto de los métodos mostraron dar buenos resultados al filtrar las mejores variables para el algoritmo. En lo que resta de este trabajo se usará solamente la

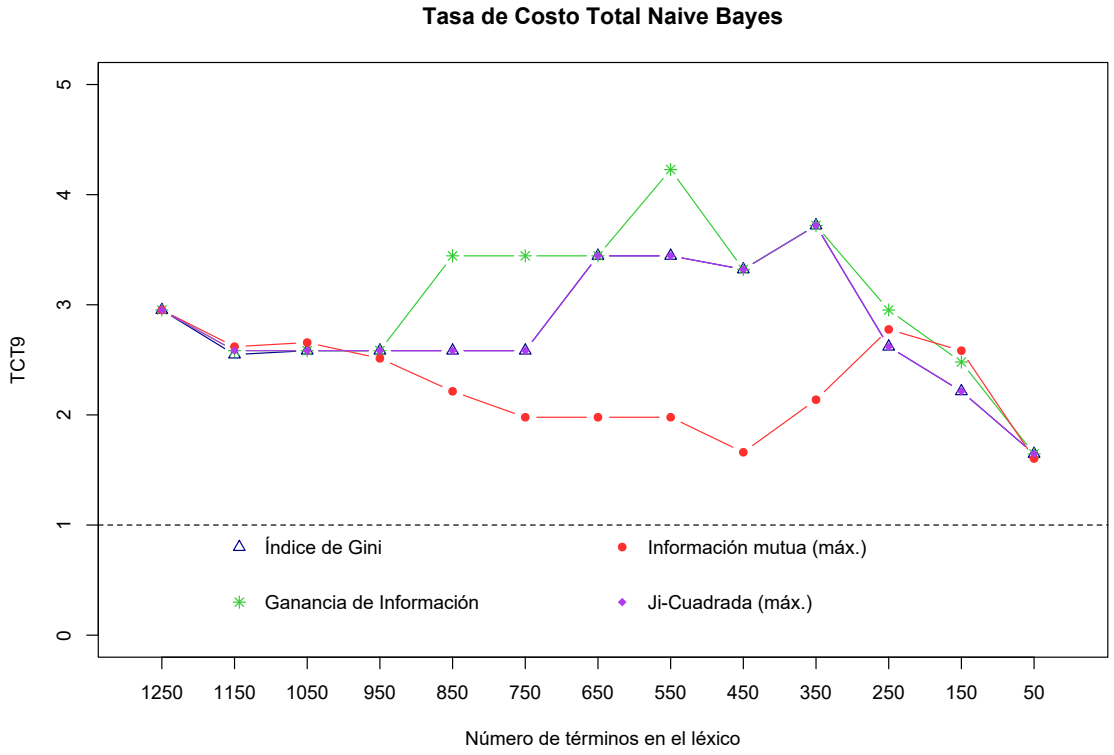


Figura 5.6: Tasa de Costo total con $\lambda = 9$ para el clasificador Naive Bayes sin suavizamiento Laplaciano usando varios métodos de selección de variables.

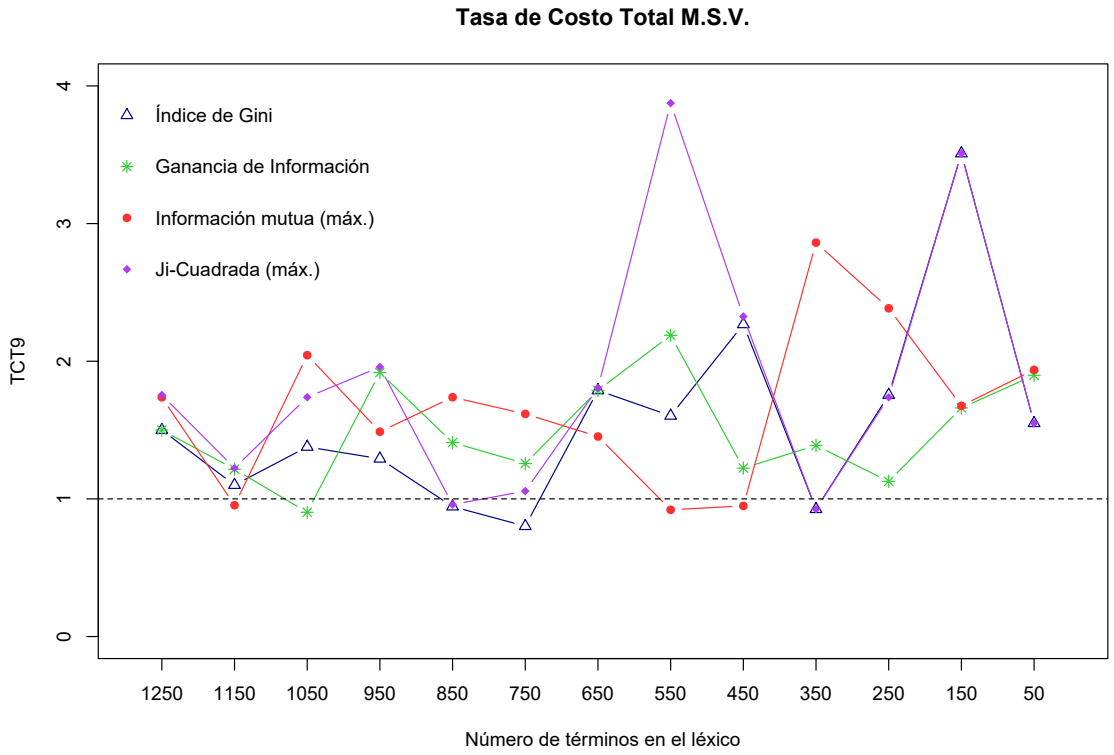


Figura 5.7: Tasa de Costo total con $\lambda = 9$ para una máquina de soporte vectorial con un kernel lineal usando varios métodos de selección de variables.

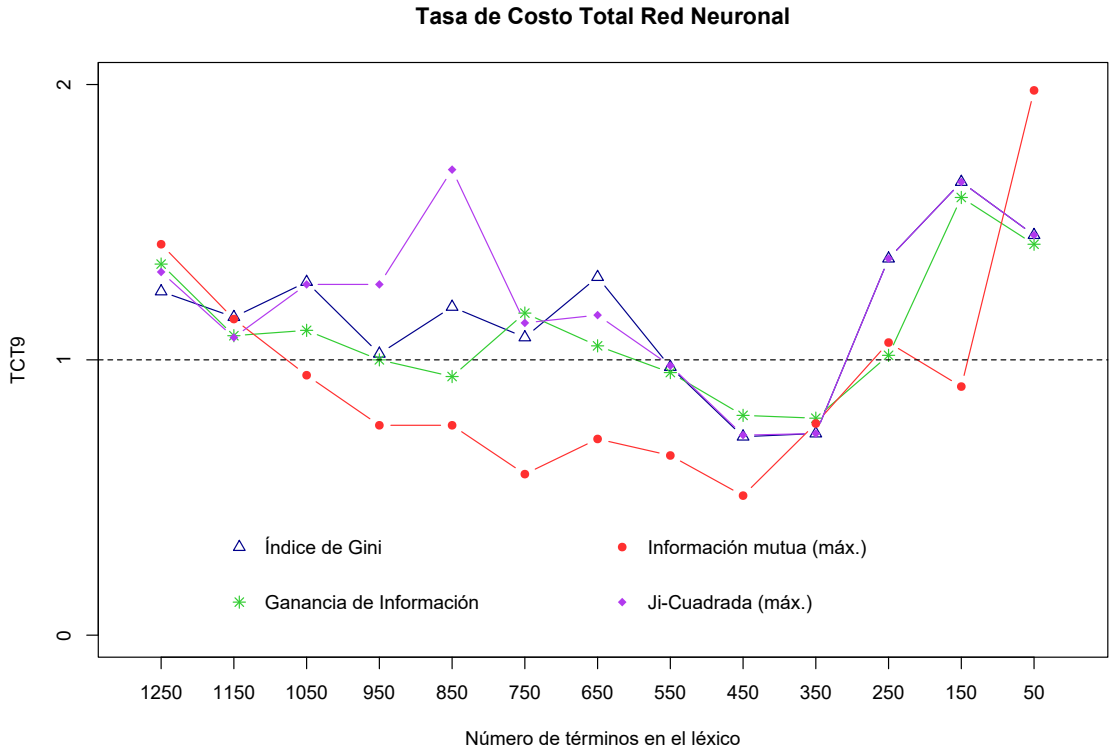


Figura 5.8: Tasa de Costo total con $\lambda = 9$ para una red neuronal sin ninguna capa oculta (perceptrón) usando varios métodos de selección de variables.

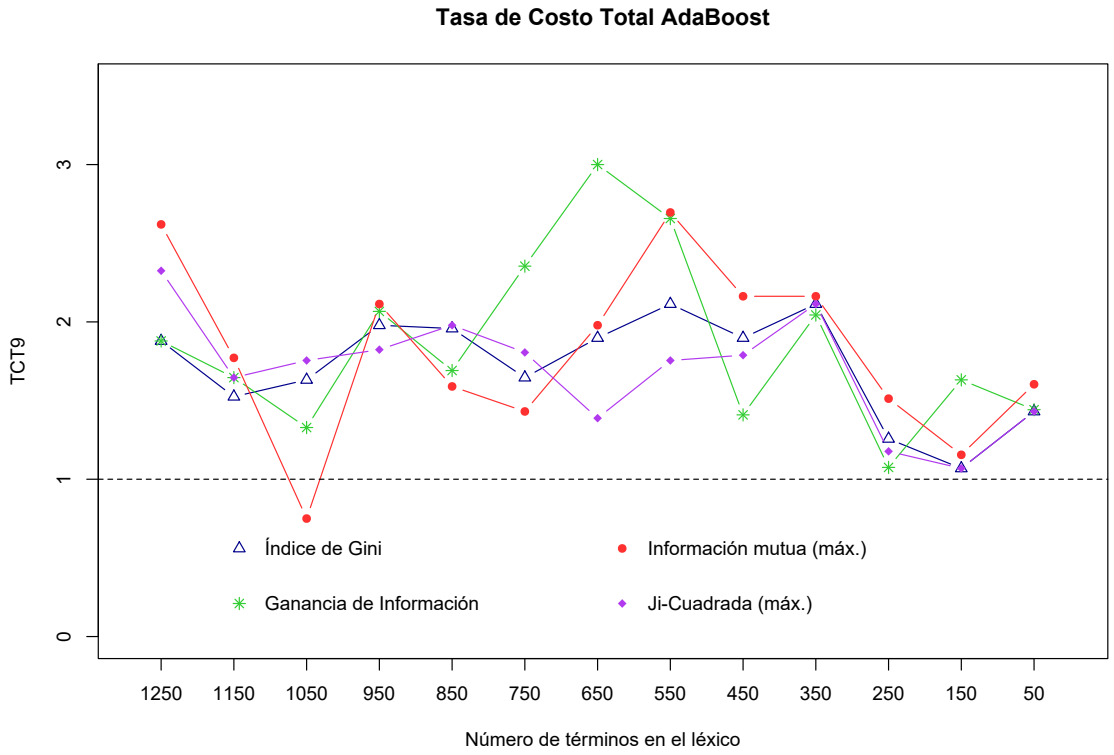


Figura 5.9: Tasa de Costo total con $\lambda = 9$ para el algoritmo AdaBoost con 30 rondas de boosting y topes de decisión como clasificadores base usando varios métodos de selección de variables.

medida de ganancia de información como método de selección de variables ya que su efectividad fue corroborada en las investigaciones realizadas y es una de las métricas más comúnmente usadas en las aplicaciones de este tipo (véase [65], [23]).

5.4.2. SVM vs Kernel SVM

Las investigaciones de algunos autores sugieren que no es necesario aplicar funciones kernel en la implementación de las máquinas de soporte vectorial para obtener resultados competitivos en el problema de la detección de SPAM y en general en la categorización de documentos (véase [23] y [3] respectivamente), sin embargo los resultados obtenidos en la gráfica 5.7 no son del todo gratos. Con el fin de buscar incrementar los valores de la tasa de costo total se han decidido explorar otras opciones de funciones kernel que se ajusten mejor a la topología particular de estos datos.

Las funciones kernel elegidas pertenecen a la familia radial, polinomial y sigmoidal (véase sección 3.3). Para cada elección de kernel se probaron algunas combinaciones de los parámetros y se midió su despemeño en la muestra de entrenamiento usando validación cruzada con 5 *folds*. Dado el alto costo computacional de este experimento se seleccionaron sólo aquellas combinaciones de parámetros que mostraron dar un mejor rendimiento y para algunos casos se definieron búsquedas en grids reducidos. Para la familia sigmoidal se exploró un grid de 9 combinaciones para los parámetros σ y C. Los parámetros de escala b y el intercepto c en la familia polinomial se fijaron en 0.001 y 0 respectivamente. El grado d y el costo C se exploraron en un grid de tamaño 6. El parámetro de escala a en el kernel sigmoidal se determinó de acuerdo al número de variables seleccionadas como $(\text{Número de variables})^{-1}$ y el intercepto b se fijó en cero. El parámetro de costo C se exploró en un grid de tamaño 5. Para el kernel lineal se consideraron 6 posibles valores del costo. En cada caso el mejor modelo fue elegido de acuerdo a su desempeño medido a través del área bajo la curva ROC.

Para $\lambda = 1$ los resultados obtenidos muestran un interesante comportamiento. Con un kernel radial o uno sigmoidal se obtiene un desempeño incluso peor que con un kernel lineal (figura 5.10). La situación es diferente cuando un kernel polinomial es aplicado. La figura 5.10 muestra que un incremento significativo en la tasa de costo total puede ser alcanzado cuando un kernel polinomial es aplicado. Esto puede ser explicado porque un kernel polinomial no es mas que una generalización de uno lineal al tomar $b, d = 1$ y $c = 0$. Esto permite que el kernel se ajuste suavemente a los datos sin incurrir en problemas de sobre-ajuste.

Aunque asumiendo costos de mala clasificación distintos el kernel polinomial no se ve tan beneficiado como en el caso anterior, sigue siendo mejor usar otra elección de kernel en vez del lineal. En este caso las mejores opciones resultan ser el kernel sigmoidal y el polinomial. El mejor desempeño lo reportó el kernel sigmoidal con un léxico de 950 palabras.

Este experimento tuvo como objetivo evaluar la influencia de las funciones kernel en el diseño de una máquina de soporte vectorial. Los resultados obtenidos muestran que mejoras significativas pueden ser alcanzadas al considerar distintas funciones kernel. Para $\lambda = 1$ el mejor modelo fue el que incluía una función kernel polinomial, mientras que con $\lambda = 9$ el mejor desempeño fue alcanzado con un kernel sigmoidal. Los siguientes experimentos se realizarán considerando los modelos ganadores de esta sección.

5.4.3. Perceptrón vs. perceptrón multicapa

Los resultados obtenidos con las máquinas de soporte vectorial en la sección anterior nos motivan a pensar que un modelo más complejo de red Neuronal puede ofrecer mejores resultados. Dada la alta dimensionalidad del problema se optó por considerar una red neuronal de una capa oculta con número reducido de unidades ocultas. También se introdujo un término de regularización para penalizar valores grandes en los pesos de la red. Los parámetros en el modelo se determinaron por validación cruzada con 5 *folds* y se exploró un grid de tamaño 9 obtenido de las combinaciones de los valores 1, 3 y 5 para el número de unidades ocultas y 0, 0.1 y 0.0001 para el término de regularización.

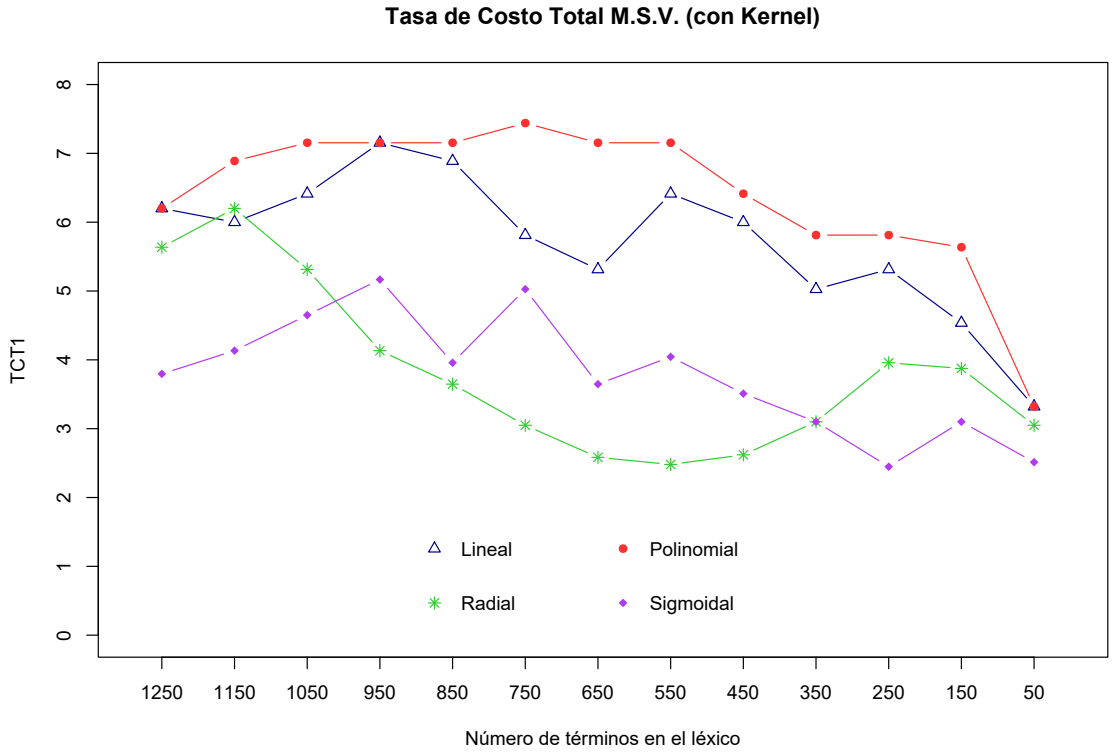


Figura 5.10: Tasa de Costo total con $\lambda = 1$ para una máquina de soporte vectorial entrenada con diferentes funciones kernel.

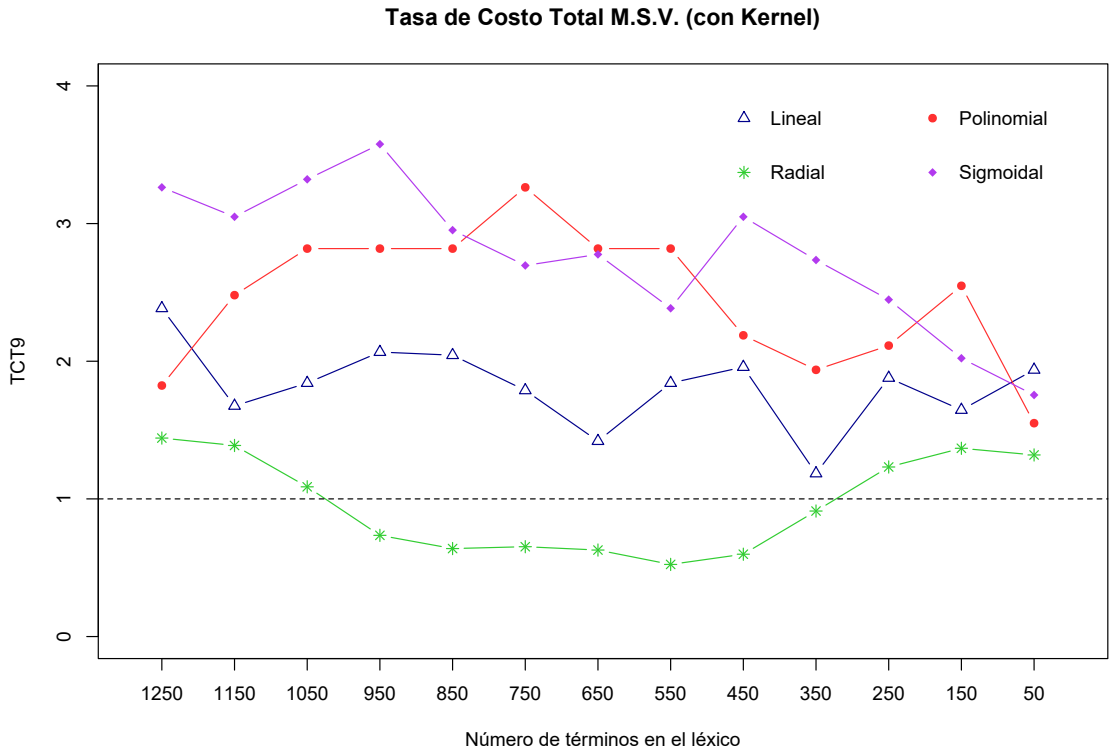


Figura 5.11: Tasa de Costo total con $\lambda = 9$ para una máquina de soporte vectorial entrenada con diferentes funciones kernel.

Los resultados obtenidos para $\lambda = 1$ muestran una clara mejoría al optar por una red neuronal con una capa oculta y un término de regularización (figura 5.12). Después de 250 términos en el léxico el rendimiento en el perceptrón multicapa supera al del perceptrón. Esto prueba que las no linealidades inducidas por la red neuronal al considerar una capa oculta y la adición de un término de regularización a la función del error son suficientes para obtener un modelo más competitivo.

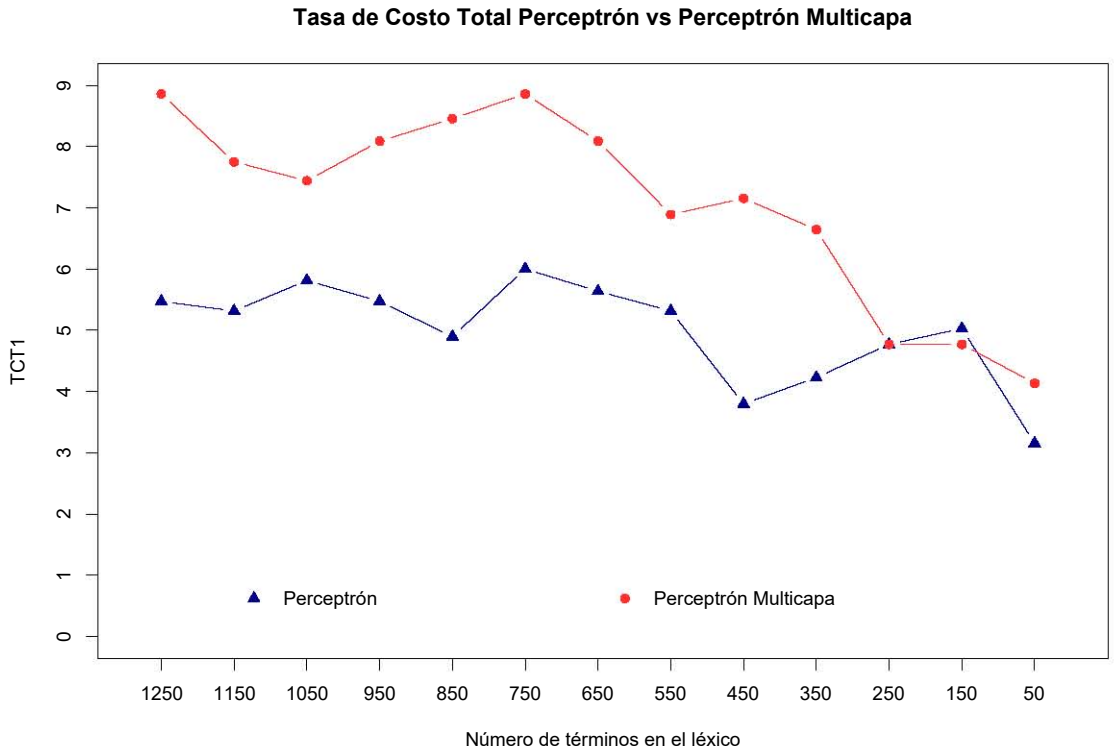


Figura 5.12: Tasa de Costo total con $\lambda = 1$ para una red neuronal entrenada con una capa oculta (perceptrón multicapa) y ninguna (perceptrón).

El escenario producido al considerar $\lambda = 9$ es aún más motivador. Mientras que el perceptrón ofrece un rendimiento equiparable al caso base de la tasa de costo total, i.e. un filtro de SPAM que no hace absolutamente nada, el perceptrón multicapa muestra valores que están siempre por encima de la recta $y = 1$, obteniendo los mejores valores para la tasa de costo total con 750 variables y 1250 (figura 5.13). Por parsimonia se seleccionaría en este caso un perceptrón multicapa entrenado con 750 variables. En lo que resta de este análisis se tomarán en cuenta sólo los mejores modelos obtenidos en este experimento.

5.4.4. Evaluación de los clasificadores

Esta sección tiene como objetivo presentar los resultados finales obtenidos de las secciones anteriores y hacer una comparativa global de los algoritmos en términos de la tasa de costo total y sus propiedades algorítmicas. Los mejores algoritmos encontrados en las secciones 5.4.2 y 5.4.3 fueron considerados en este análisis.

Para $\lambda = 1$ la figura 5.14 muestra los resultados obtenidos por los 4 clasificadores. Se puede notar que la red neuronal con una capa oculta y un término de regularización y la máquina de soporte vectorial con un kernel polinomial ofrecen los mejores resultados. El clasificador Naive Bayes y el algoritmo AdaBoost también ofrecen resultados competitivos aunque no tan buenos como los anteriores. En particular la red neuronal presenta en este caso el mejor desempeño global con 750 palabras en el léxico. En general se observa que el desempeño de los algoritmos tiene una tendencia creciente al incrementar el número de palabras y se estabilizan después

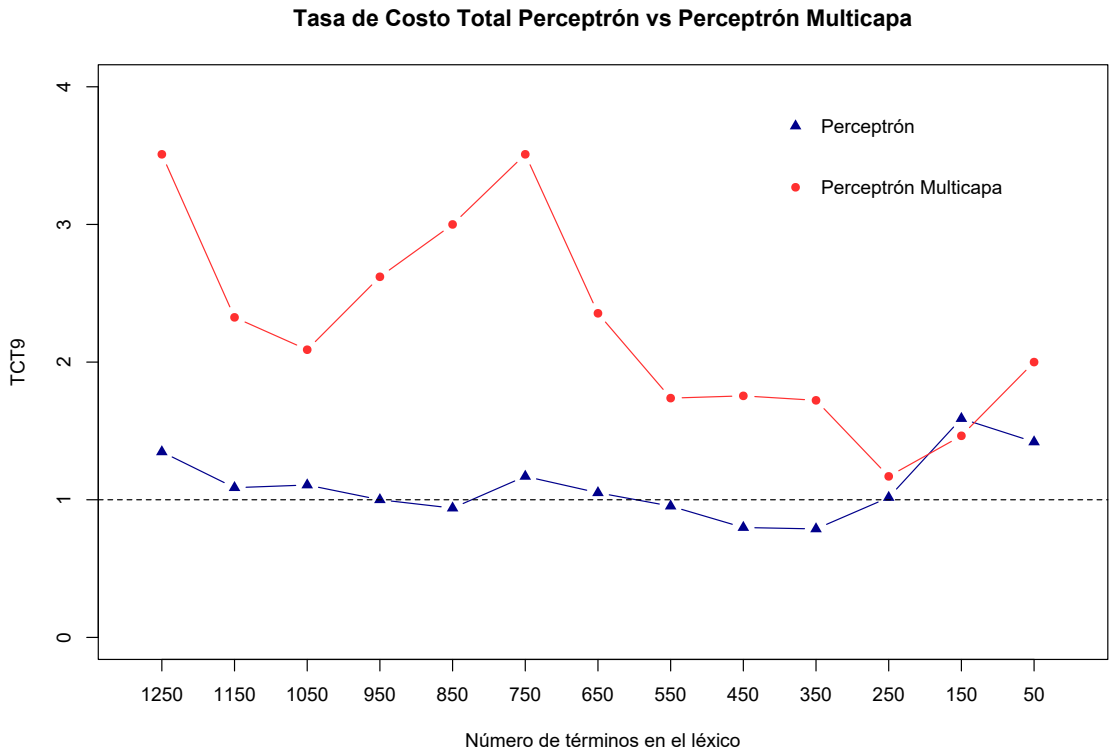


Figura 5.13: Tasa de Costo total con $\lambda = 9$ para una red neuronal entrenada con una capa oculta (perceptrón multicapa) y ninguna (perceptrón).

de 550 palabras.

Al considerar costos de mala clasificación distintos ($\lambda = 9$) el panorama es diferente y en este caso el clasificador Naive Bayes reporta la mejor tasa de costo total con 550 variables (figura 5.15). En este caso la tendencia creciente se ha perdido y los algoritmos muestran un comportamiento más plano al incrementar el número de palabras. Un hecho motivador es que en este caso cualquiera de los algoritmos reporta un buen desempeño al estar todos por encima de la recta $y = 1$. En este caso la máquina de soporte vectorial con un kernel sigmoidal y el perceptrón multicapa con una capa oculta y un término de regularización también muestran resultados competitivos. El peor desempeño lo presenta el algoritmo AdaBoost en la mayoría de los casos.

Las figuras 5.14 y 5.15 muestran un total de 104 posibles modelos obtenidos al considerar diferentes tamaños de léxicos. 52 de ellos se obtienen al tomar $\lambda = 1$ y el resto con $\lambda = 9$. Con el fin de simplificar el análisis, para cada algoritmo se ha seleccionado el tamaño de léxico que maximiza la tasa de costo total. Las tablas 5.6 y 5.7 muestran un resúmen de los algoritmos seleccionados, su tamaño de léxico óptimo así como sus parámetros y su tasa de costo total.

Resumen de algoritmos seleccionados TCT1			
Algoritmo	Tamaño del léxico	Parámetros	TCT1
Naive Bayes	550	Suavizamiento Laplaciano: No	6.643
MSV (polinomial)	750	$b = 0.001$, $c = 0$, $d = 1$, $C = 0.25$	7.440
Red neuronal	750	unidades ocultas=3, regularización=0.1	8.857
AdaBoost	550	Rondas=30, profundidad del árbol=1	6.200

Tabla 5.6: Resúmen de los modelos seleccionados para la TCT con $\lambda = 1$

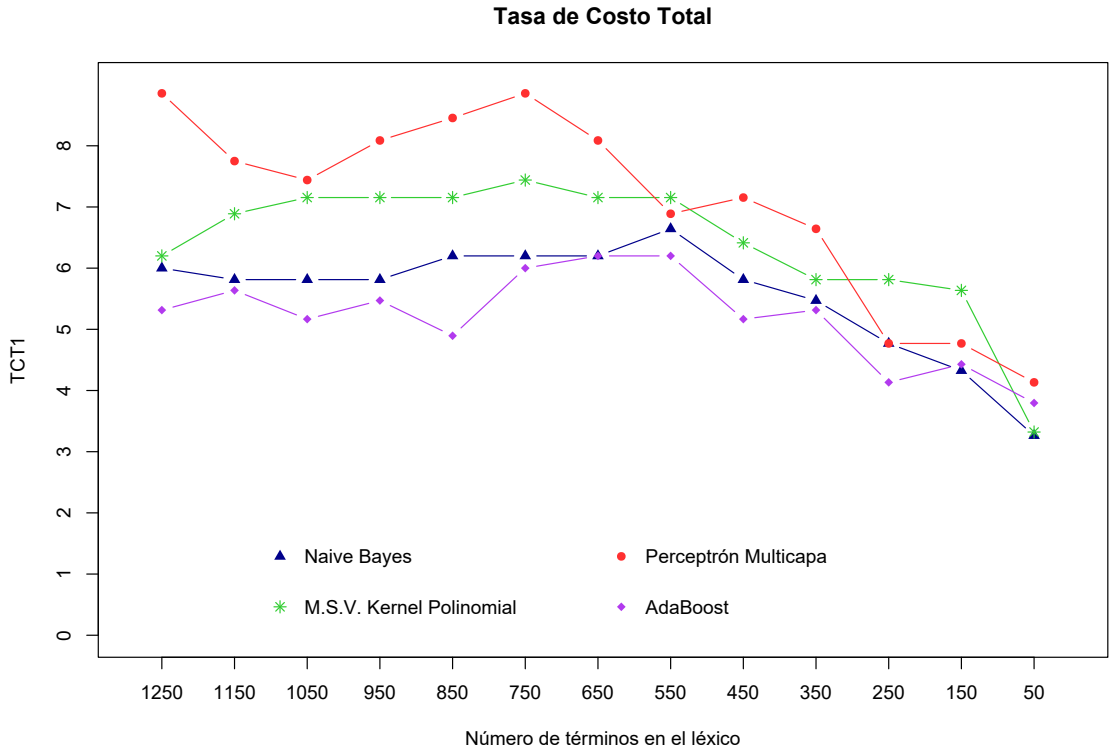


Figura 5.14: Tasa de Costo total con $\lambda = 1$ para diferentes algoritmos.

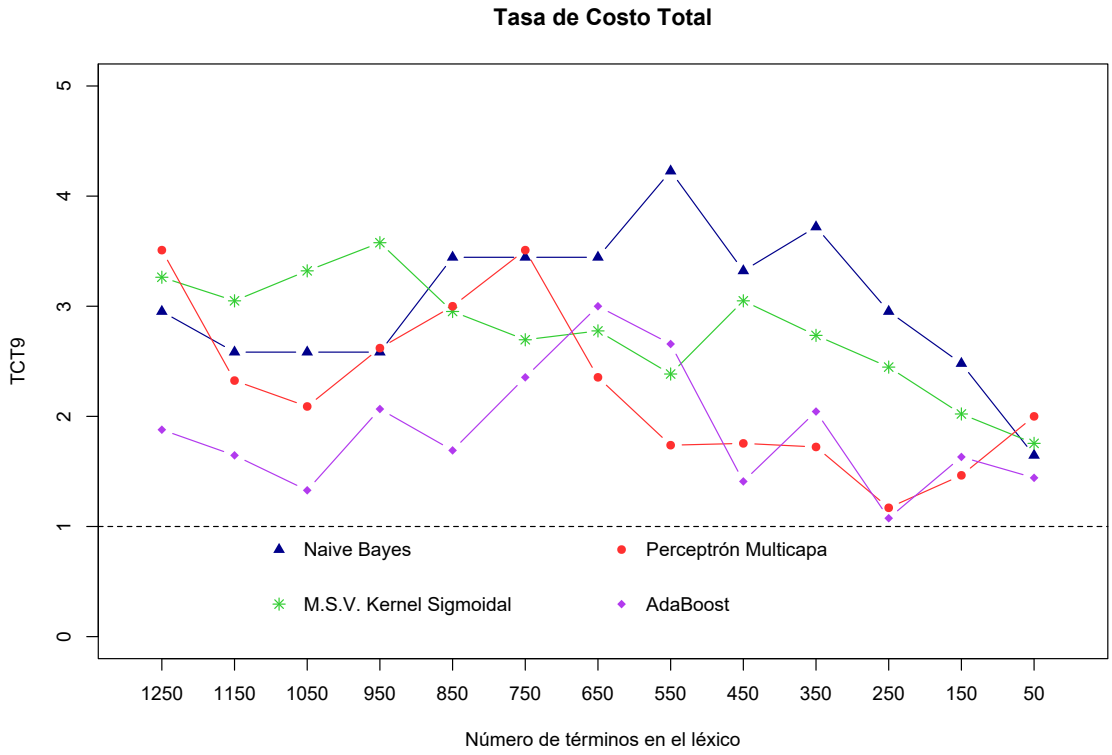


Figura 5.15: Tasa de Costo total con $\lambda = 9$ para diferentes algoritmos.

Resumen de algoritmos seleccionados TCT9			
Algoritmo	Tamaño del léxico	Parámetros	TCT9
Naive Bayes	550	Suavizamiento Laplaciano: No	4.227
MSV (sigmoidal)	950	$a = 0.001052632$, $b = 0$, $C = 0.5$	3.577
Red neuronal	750	unidades ocultas=3, regularización=0.1	3.509
AdaBoost	650	Rondas=30, profundidad del árbol=1	3.000

Tabla 5.7: Resúmen de los modelos seleccionados para la TCT con $\lambda = 9$

Con el fin de obtener conclusiones estadísticas se ha realizado un bootstrap paramétrico de 100 iteraciones en la muestra de prueba estratificando por tipo de mensaje para cada uno de los algoritmos seleccionados que se muestran en las tablas 5.6 y 5.7. De esta manera se ha obtenido una aproximación a la distribución de la tasa de costo total para $\lambda = 1$ y $\lambda = 9$. Los resultados se muestran en la figura 5.16. A partir de esta distribución es posible construir intervalos de confianza y estimar otras características poblacionales para la tasa de costo total.

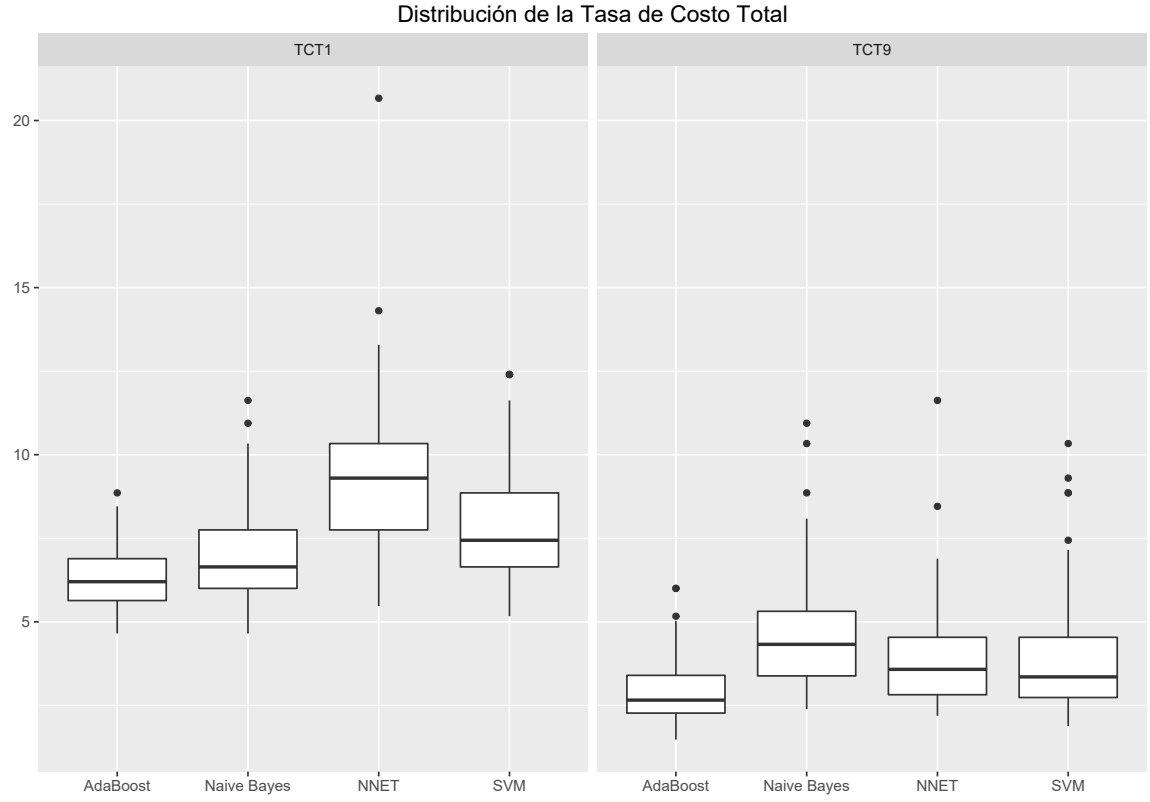


Figura 5.16: Distribución de la TCT en la muestra de prueba para los algoritmos con los mejores conjuntos de variables seleccionados. Los resultados fueron obtenidos aplicando un bootstrap con 100 iteraciones.

La figura 5.17 muestra un intervalo aproximado para la tasa de costo total obtenido al calcular los percentiles 0.025 y 0.975 de la distribución generada con el bootstrap para cada algoritmo. Otras características como la media o la varianza pueden ser obtenidas de la misma manera.

Una manera de probar si existe diferencia entre los algoritmos seleccionados es contrastando su rendimiento medio mediante una prueba de hipótesis. Aunque algunos trabajos usan la prueba t para la diferencia de medias entre dos poblaciones, en este caso se ha optado por la prueba de suma de rangos de Wilcoxon [70] con el fin de evitar el supuesto de normalidad y los algoritmos han sido comparados según sus distribuciones en la

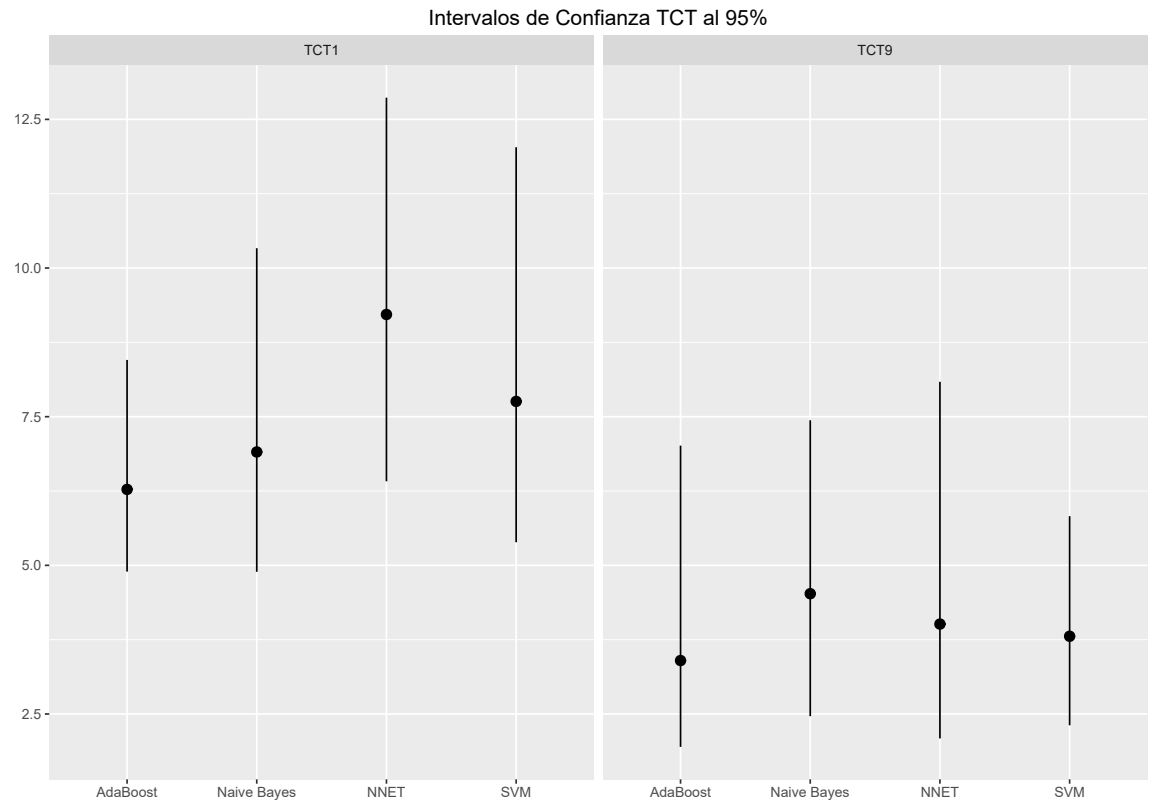


Figura 5.17: Intervalos al 95 % de confianza para la TCT en la muestra de prueba para los algoritmos con los mejores conjuntos de variables seleccionados. Los resultados fueron obtenidos aplicando un bootstrap con 100 iteraciones.

tasa de costo total con $\lambda = 1$ y $\lambda = 9$ obtenidas del bootstrap. Los resultados de la prueba se encuentran en las tablas 5.8 y 5.9. Aunque es posible realizar un total de 12 comparaciones (6 por cada valor de lambda), por simplicidad sólo se muestran las 6 más interesantes según los comportamientos que revelan las gráficas 5.14 y 5.15.

Los resultados de la prueba de Wilcoxon para $\lambda = 1$ muestran que existe evidencia estadística para afirmar que en promedio el rendimiento de todos los algoritmos es diferente al rechazar la hipótesis nula que afirma igualdad de medias con *p-values* menores a un nivel de significancia del 0.05.

Para $\lambda = 9$ la prueba no se rechaza cuando se comparan la red neuronal contra la máquina de soporte vectorial (nótese que en este caso el *p-value* es mayor a 0.05 o equivalentemente el intervalo al 95 % confianza no contiene al cero). Esto significa que no existe evidencia estadística para determinar un ganador entre ambos

Resumen de la prueba de suma de rangos de Wilcoxon ($TCT = 1$)			
Comparación	Intervalo al 95 % de Confianza	*p-value*	¿Rechazar H_0 : ?
NNET-SVM	$(0.932, 1.887)$	0.0000000077	SI
SVM-Naive Bayes	$(0.387, 1.154)$	0.0000089370	SI
Naive Bayes-AdaBoost	$(0.214, 0.797)$	0.0007435000	SI

Tabla 5.8: Resumen de la prueba de suma de rangos de Wilcoxon (H_0 : El desempeño medio (TCT1 media) de los modelos es el mismo) para la TCT con $\lambda = 1$ considerando los algoritmos con los mejores rendimientos. Se ha tomado $\alpha = 0.05$.

Resumen de la prueba de suma de rangos de Wilcoxon ($TCT = 9$)			
Comparación	Intervalo al 95 % de Confianza	*p-value*	¿Rechazar H_0 : ?
Naive Bayes-SVM	$(0.314, 0.918)$	0.0001159000	SI
SVM-NNET	$(-0.258, 0.357)$	0.7231000000	NO
NNET-AdaBoost	$(0.207, 0.798)$	0.0006932000	SI

Tabla 5.9: Resumen de la prueba de suma de rangos de Wilcoxon (H_0 : El desempeño medio (TCT9 media) de los modelos es el mismo) para la TCT con $\lambda = 9$ considerando los algoritmos con los mejores rendimientos. Se ha tomado $\alpha = 0.05$.

algoritmos. El resto de los modelos reportan diferencias en sus rendimientos promedio al rechazar la hipótesis nula con un nivel de significancia de 0.05.

De esta sección se concluye que con $\lambda = 1$ los algoritmos muestran desempeños competitivos entre sí, siendo en este caso la red neuronal con 3 unidades ocultas, un término de regularización de 0.1 y 750 variables la que mostró el mejor resultado al alcanzar una tasa de costo total de 8.857. Para el caso $\lambda = 9$ los algoritmos no muestran una tendencia tan clara como en el caso anterior y es difícil determinar un ganador, sin embargo el mejor rendimiento lo registró el clasificador naive Bayes sin suavizamiento Laplaciano con 550 variables al alcanzar una tasa de 4.227.

5.4.5. MSV vs MRV

Este sencillo experimento tiene como objetivo comparar la efectividad de las máquinas de soporte vectorial contra su versión Bayesiana, las máquinas de relevancia vectorial. Con el fin de hacer más ilustrativo este experimento se ha reducido la dimensión de la base de datos *SPAM E-mail Database* a dos dimensiones usando la descomposición en valores singulares del análisis semántico latente presentado en la sección 2.5. Así, cada correo electrónico ha sido clasificado según los valores reportados por sus dos tópicos latentes.

Para realizar el experimento se han entrenado una máquina de soporte vectorial y una máquina de relevancia vectorial. Para ambos modelos se ha usado un kernel radial. En el caso de la MSV el parámetro de costo C ha sido determinado vía validación cruzada al considerar un grid reducido y los parámetros asociados a la máquina de relevancia vectorial han sido determinados de manera implícita en la verosimilitud del modelo. La figura 5.10 muestra las matrices de confusión obtenidas al entrenar ambos modelos, en ellas se observa que los resultados son similiares. Las curvas ROC y el área bajo la curva también han reportado rendimientos similares (véase figura 5.20)

La dimensión reducida inducida por los tópicos latentes permite visualizar en dos dimensiones la manera en que operan los algoritmos. Las figuras 5.18 y 5.19 muestran las fronteras de decisión definidas por los modelos. Se puede observar que la elección de funciones kernel permite que los modelos adapten sus fornteras de decisión a la topología de los datos. Las figuras también muestran los vectores de soporte y de relevancia obtenidos para cada modelo respectivamente. La máquina de soporte vectorial definió 1266 vectores de soporte y la de relevancia tan sólo 3 vectores de relevancia. Esto prueba que las máquinas de relevancia vectorial pueden producir resultados robustos y eficientar al mismo tiempo el costo computacional al operar con un número menor de funciones kernel.

La tabla 5.11 muestra un resúmen de los resultados obtenidos en el experimento. Se puede notar que en términos de la tasa de costo total y la exactitud, ambos algoritmos alcanzan rendimientos comparables. Aunque esto era de esperarse, dadas los resultados obtenidos en las matrices de confusión (tabla 5.10), la verdadera sorpresa surge al comparar el número de funciones kernel y el tiempo de entrenamiento requerido. Para el

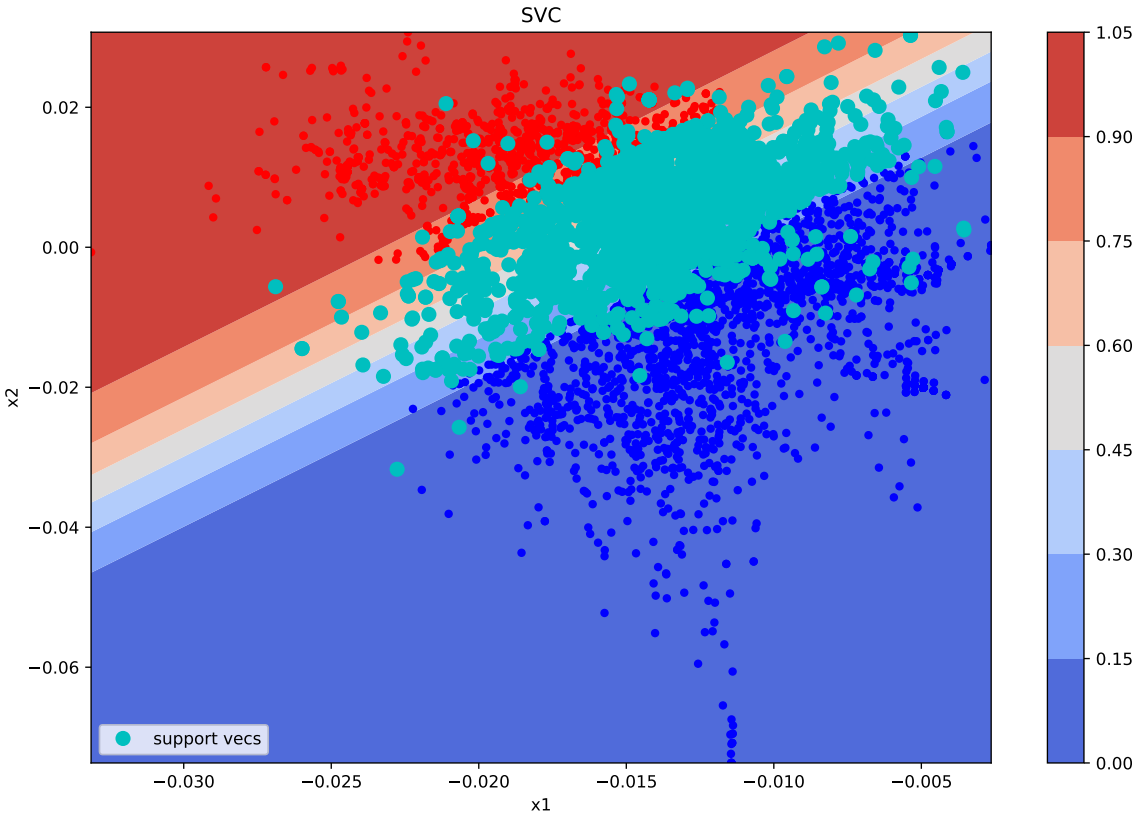


Figura 5.18: Se muestra la frontera de decisión definida por una máquina de soporte vectorial en la base de datos *SPAM E-mail Database* con dos tópicos latentes.

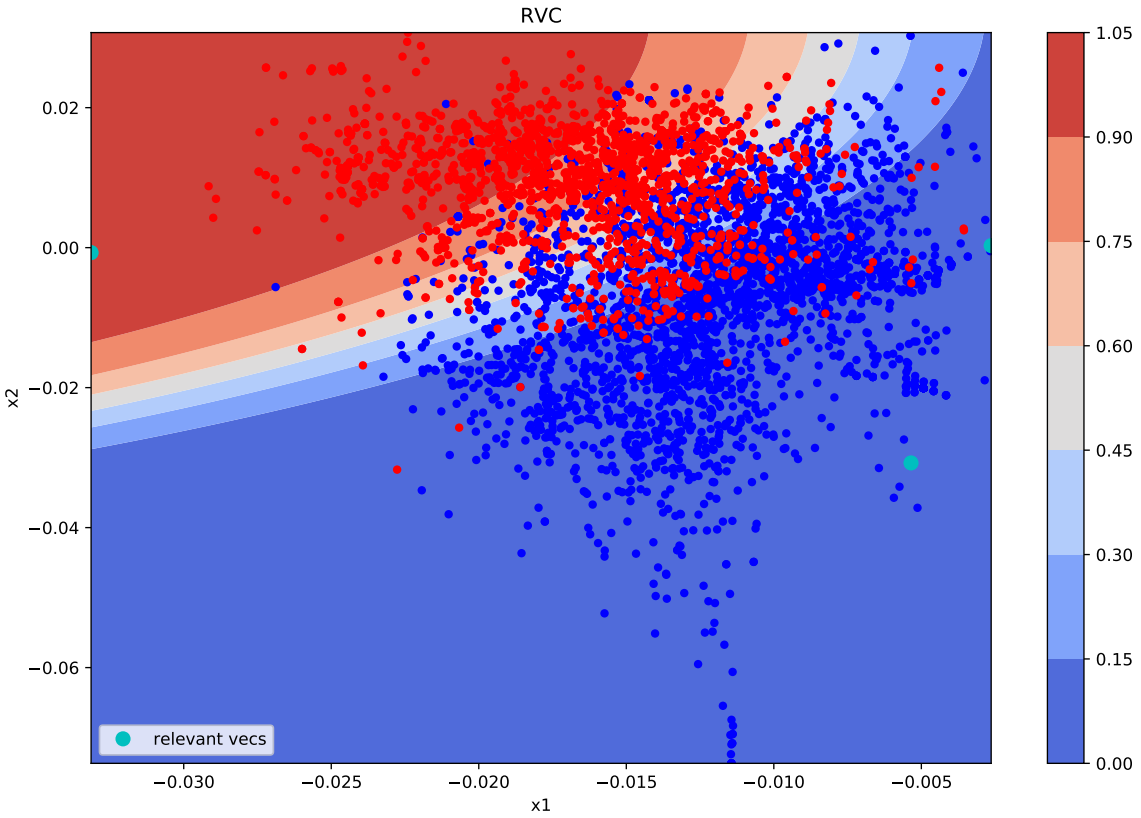


Figura 5.19: Se muestra la frontera de decisión definida por una máquina de relevancia vectorial en la base de datos *SPAM E-mail Database* con dos tópicos latentes.

Valor actual	Predicción: spam	Predicción: no spam	Total
spam	1071	267	168
no spam	227	1885	1225
Total	187	1206	

Valor actual	Predicción: spam	Predicción: no spam	Total
spam	1075	263	1338
no spam	233	1879	2112
Total	1308	2142	

Tabla 5.10: A la izquierda (derecha) se muestra la matriz de confusión obtenida por una máquina de soporte (relevancia) vectorial en la base de datos *SPAM E-mail Database*

Resúmen MSV vs MRV (*SPAM E-mail Database*)					
Model	TCT1	TCT9	Exactitud	Kernels	Tiempo de entrenamiento
MSV	2.708	0.579	86.62 %	1266	56.12s
MRV	2.697	0.567	86.79 %	3	17.47s

Tabla 5.11: Resúmen de los resultados obtenidos en el experimento MSV vs MRV en la base *SPAM E-mail Database*

caso de las máquinas de soporte vectorial hace falta entrenar el modelo un número repetido de veces para poder determinar los parámetros por validación cruzada o alguna otra técnica de remuestreo y esto hace que el tiempo total consumido en la fase de entrenamiento incremente rápidamente. Por el contrario, las máquinas de relevancia vectorial estiman todos sus parámetros directamente de la verosimilitud y ello hace que, junto con algún método de optimización, se puedan estimar los parámetros en un periodo relativamente corto de tiempo. La drástica reducción en el número de funciones kernel también indica que resultados significativamente más escasos pueden ser producidos por las máquinas de relevancia vectorial.

5.5. Conclusiones

Los experimentos en la base de datos *SMS SPAM Collection* tuvieron como objetivo investigar y comparar algunas de las técnicas de clasificación más usadas en minería de textos y al mismo tiempo ejemplificar el preprocesamiento de datos requerido bajo este tipo de contextos. Vale la pena hacer algunas observaciones al respecto del desempeño y las características de cada modelo durante el experimento.

- Naive Bayes: Este algoritmo es un claro ejemplo de que una idea sencilla puede resolver un problema complicado. En este caso la estructura de correlación en el léxico parece beneficiar al supuesto de independencia del algoritmo, aunque esto puede no ser así en muchos otros casos. En [23] se prueba que este suspuesto es demasiado restrictivo para una base de caracteres chinos. Otra de las ventajas de este algoritmo es su bajo costo computacional. Dado que sólo requiere estimar un conjunto de probabilidades condicionales, el algoritmo puede operar con un gran número de datos y variables de manera eficaz. Aunque con $\lambda = 1$ el algoritmo no mostró los mejores resultados, para $\lambda = 9$ probó ser uno de los mejores.

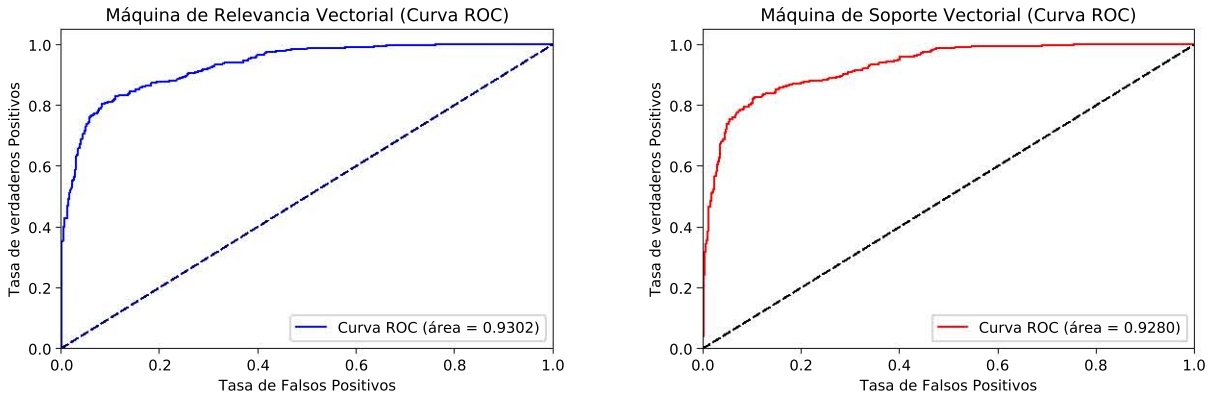


Figura 5.20: A la izquierda (derecha) se muestra la curva ROC y su área para la máquina de relevancia (soporte) vectorial en la base de datos SPAM.

- Máquinas de Soporte Vectorial: Las máquinas de soporte vectorial han mostrado ser uno de los algoritmos de aprendizaje supervisado más efectivos en tareas de clasificación de texto. En este trabajo han mostrado resultados muy competitivos viéndose superadas en algunos casos sólo por la red neuronal. Su tiempo de entrenamiento es relativamente corto en problemas de alta dimensionalidad y por ello es una de las herraminetas más usadas en este tipo de contextos. Una desventaja importante es el número de parámetros libres en el modelo, que en el peor de los casos podrá ser de hasta 3 o 4 dependiendo de la función kernel elegida. Por otro lado las MSV se ven beneficiadas de una rica teoría y la noción de márgen de máxima separación es intuitiva y proporciona más claridad sobre el funcionamiento del algoritmo.

  La posibilidad de introducir una función kernel hace que el algoritmo goce de mucha flexibilidad. Un enfoque adoptado recientemente consiste en cambiar la modelación típica de los documentos al introducir un *kernel de cadena* [71]. Bajo este nuevo esquema los documentos pueden ser vistos como "cadenas de caracteres" en vez de "bolsas de palabras".

- Redes Neuronales: A pesar del escepticismo que las rodea, las redes neuronales han probado ser uno de los modelos de aprendizaje más efectivos. Un entrenamiento cuidadoso de la red puede producir los mejores resultados. Su desventaja más evidente es quizás el elevado número de parámetros que surgen en problemas de alta dimensionalidad. En este experimento el perceptrón de una capa oculta con un término de regularización mostró dar los mejores resultados globales en comparación con el resto de los algoritmos para $\lambda = 1$, sin embargo el tiempo de entrenamiento fue considerablemente más elevado y ello puede resultar restrictivo en léxicos más grandes.

  Una de las críticas más severas hacia este tipo de modelos es su poca o nula capacidad interpretativa y por ello son considerados en muchos casos dentro de la categoría de "algoritmos tipo caja negra". Una posible solución a este problema es la construcción de un modelo sustituto. Un modelo sustituto es un algoritmo de aprendizaje a través del cual se pretende explicar el comportamineto de alguno otro. Por ejemplo, es posible construir un árbol de decisión que aproxime a la red neuronal para explicar así el comportamiento de la red en términos de las decisiones del árbol. Otras de las desventajas de este modelo son la no unicidad, dada por la no convexidad en la función de error, y los problemas de sobreajuste provocados por largos periodos de entrenamiento. A pesar de lo anterior, el éxito de este modelo durante el experimento puede ser explicado por su flexibilidad y sus capacidades de generalización en problemas que involucran fronteras de decisión no lineales.

- AdaBoost: El algoritmo AdaBoost mostró resultados competitivos en la mayoría de los casos, sin embargo el costo computacional requerido al usar árboles de decisión como clasificadores base resultó prohibitivo en la realización del experimento. A pesar de que en este caso se usaron topes de decisión como clasificadores base, el costo computacional requerido (principalmente en los léxicos más grandes)

dificultó una exploración más profunda de algunos parámetros como el número de rondas de boosting o la profundidad del árbol. En este caso se decidió usar un número máximo de 30 rondas de boosting porque después de este valor no se observaron mejoras importantes en el desempeño del algoritmo pero sí un incremento substancial en el tiempo de entrenamiento para la mayoría de los léxicos. Esto revela que bajo este contexto la debilidad más importante de este algoritmo es el tiempo de entrenamiento. Otras variaciones de este algoritmo también han sido exploradas por algunos autores. En [23] por ejemplo, se puede encontrar un estudio que contempla el algoritmo LogiBoost.

El experimento realizado en la base *SPAM E-mail Database* tuvo como objetivo evidenciar las diferencias y similitudes entre las máquinas de soporte y las máquinas de relevancia vectorial. Este experimento permitió confirmar muchas de las afirmaciones que algunos estudios respecto al tema han concluido (véase [44], [72]). Los resultados más importantes obtenidos en este experimento muestran que las máquinas de relevancia vectorial permiten aumentar en un 99.8 % la escasez del modelo y reducir en un 68.9 % el tiempo de entrenamiento (estos valores fueron obtenidos al dividir el número de vectores de relevancia entre el número de vectores de soporte y el tiempo de entrenamiento medido en segundos de cada algoritmo respectivamente). Esto prueba los beneficios de la modelación Bayesiana en este contexto. Las conclusiones sobre la exactitud y la tasa de costo total para cada algoritmo también son muy similares y esto prueba que resultados parecidos y con un menor costo computacional pueden ser alcanzados por una máquina de relevancia vectorial. Otro punto a favor del enfoque Bayesiano es que éste permite obtener salidas probabilísticas en vez de medidas de cercanía a una frontera, esto puede tener más sentido y ser de utilidad en contextos donde es necesario cuantificar la incertidumbre en el proceso de clasificación.

Capítulo 6

Análisis de sentimientos

La influencia de las redes sociales en las personas ha cobrado tanto impacto que las organizaciones han desarrollado una preocupación particular por monitorear y analizar su comportamiento. Desde opiniones y críticas hasta sentimientos y emociones son plasmados en las redes sociales día a día por las personas en todo el mundo. Son tantos los fines para los que puede usarse una red social que algunos de ellos pasan inadvertidos e incluso impactan directamente en nuestras vidas sin siquiera darnos cuenta. Algunos analistas atribuyen la victoria de Barack Obama en Estados Unidos a su extensa campaña en línea. Sitios no oficiales como Wikipedia publican artículos como "Barack Obama on social media" en los que describen el impacto del presidente durante su campaña en las redes sociales. La revista "The Atlantic" lo ha citado incluso como "the first social-media president". En respuesta a este fenómeno estudios serios han surgido al respecto. Andranik Tumasjan et al. [73] analizan, por ejemplo, una muestra de tweets sobre elecciones presidenciales en su artículo "Predicting Elections with Twitter". El impacto de las redes sociales en áreas como las finanzas ha sido estudiado también. El libro "Text Mining and Visualization" de Hofmann et al. [24] propone analizar el comportamiento del mercado de valores usando una muestra de tweets y el software estadístico R.

El análisis de sentimientos o minería de opinión es una novedosa área de investigación que surge en respuesta al deseo por conocer las opiniones y las tendencias que siguen las personas en las redes sociales, blogs o páginas web dedicadas a diversas actividades. El fin del análisis de esta información es muy diverso e investigaciones recientes indican que puede ser aplicado en áreas como finazas, economía, política y estudios de mercado. Bing Liu [4] define el análisis de sentimientos de la siguiente manera: "El análisis de sentimientos, también llamado minería de opinión, es el campo de estudio que analiza opiniones, sentimientos, valoraciones, actitudes y emociones de las personas hacia las entidades y sus atributos expresados en texto escrito". Hoy en día el análisis de sentimientos no está limitado solamente a las redes sociales. Muchos de los trabajos en esta área se enfocan también al estudio de las preferencias de los consumidores a partir de muestras de opiniones tomadas de sitios web especializados en ventas como "Amazon" o "mercado libre" [74], [75].

Es un hecho que las redes sociales y los medios web seguirán evolucionando trayendo consigo nueva información disponible para el análisis y la interpretación. Esto implica que hoy en día uno de los grandes retos de la estadística sea su capacidad para interactuar con otras áreas como la computación y las ciencias sociales con el fin de explotar estas nuevas plataformas. Este capítulo tiene como objetivo ejemplificar un caso particular de la minería de opiniones (o análisis de sentimientos) usando las herramientas de modelación de tópicos latentes expuestas en la sección 2.6.

6.1. Descripción de la base de datos

6.1.1. Scraping Web

Las aplicaciones más comúnes de análisis de sentimientos se enfocan al análsis de datos e información contenida en páginas o sitios web. Por lo general estos sitios web son redes sociales, blogs, foros de discusión o páginas especializadas en ventas. Una manera popular de extraer la información de estos sitios es conocida como *scraping web*, la cual forma parte de una amplia área computacional conocida como recuperación de

la información. Esta técnica consiste en programar un robot cuyo fin sea extraer la información de manera automática de estos sitios por medio de alguna aplicación con conexión a internet. Aunque no existen parámetros predeterminados para realizar este proceso, es fácil encontrar bibliotecas en la mayoría de los paquetes estadísticos que permiten facilitar la realización de esta tarea.

Cabe mencionar que aunque este paso no forma parte del análisis es igual de importante, ya que las conclusiones dependerán directamente de la manera en que hayan sido capturados los datos. Es por ello que la recolección de los datos deberá ser llevada a cabo de manera cuidadosa. Además de texto, otros metadatos pueden ser extraídos usando técnicas de *scraping web*. Por ejemplo, en una aplicación relacionada al análisis de las opiniones que un conjunto de clientes hacen respecto de un producto podría ser interesante extraer también el autor, la fecha en que fue escrita la opinión y alguna calificación que el cliente otorgue al producto (por ejemplo medida en estrellas o en una escala del 1 al 5). En una aplicación asociada a una red social como Twitter podría ser natural pensar en extraer el nombre del usuario, su ubicación geográfica y el número de veces que el mensaje fue retwitteado o el número de "me gusta" que las personas le han dado a la publicación.

6.1.2. Base de datos de Amazon

Para ejemplificar esta aplicación, fue extraída una colección de opiniones que diferentes usuarios publicaron en la página oficial de Amazon México sobre 10 productos. Para ello fueron usadas algunas técnicas de *scraping web* y se extrajeron las 100 últimas opiniones de cada producto obteniendo así una base de 1000 registros. Las descripciones de los productos considerados se encuentran a continuación.

- **Audifonos**: Sennheiser CX 300 II Precision Audifonos de boton con realce de graves, color negro
- **Bateria:** ADATA APT100 - Bateria de 10000 mAh, negro y verde
- **Cámara:** Sony DSC-H300 - Camara compacta de 20.1 MP (pantalla de 3", zoom optico 35x, estabilizador de imagen electronico, video HD 720p), negro
- **Cargador:** AUKEY Quick Charge 3.0 Cargador de Red 18W [Qualcomm Certificado] para iPhone, Nexus, ect. (Negro)
- **Teléfono:** Apple iPhone 6 16 GB, Oro Desbloqueado
- **Kindle:** Kindle Paperwhite, pantalla E-ink de alta resolución, luz integrada, color Negro, Wi-Fi
- **Memoria:** SanDisk Ultra - Tarjeta de memoria Android 32 GB microSDHC UHS-I con adaptador SD, velocidad de lectura hasta 80 MB/s y Clase 10, FFP
- **Reloj:** Smartwatch U8 Bluetooth iphone Android Reloj Inteligente
- **Televisión:** Samsung UN-32J4300AF- Television LED 32"(Smart TV)
- **Videojuego:** Halo 5: Guardians Estandar en Espanol - Xbox One - Standard Edition

Además de los comentarios que los clientes realizaban sobre los productos, un conjunto de metadatos adicionales fue extraído tales como el autor, la fecha de publicación, la calificación que el usuario otorga al producto, el título del comentario y un marcador que cuenta el número de personas para las que fue útil el comentario. Aunque para esta aplicación estos metadatos no serán utilizados, algunos autores hacen uso de ellos para ejemplificar aplicaciones de aprendizaje supervisado. Por ejemplo Callen Rain [75] propone utilizar la información proporcionada por la calificación que el cliente otorga al producto (número de estrellas o escala) para entrenar un algoritmo capaz de hacer predicciones sobre la preferencia del cliente por el producto medida en esta escala. La tabla 6.1 muestra los primeros comentarios de cada producto obtenidos como resultado de la captura de los datos.

	Producto	Comentarios
1	Audifonos Sennheiser	Ya habia tenido una par de estos y no hay ninguna queja en el apartado del sonido, que deberia se lo mas importante. Esteticamente, podrian tener un mejor diseno, sobretodo en el cable, pero como dije, no es ningun problema para su desempeno. El servicio muy bien, cumpliendo con la fecha compromiso.[...]
101	Bateria ADATA	Excelente producto en cuanto a relacion mAh/precio, busque otras opciones, en liverpool, sears, sanborns, te venden una de 10,000 mAh por encima de los 700 pesos, (mas cara si es de sony), en Mobo (tienda de accesorios de celulares) encuentras la bateria en 670 pesos y justo cuando me iba a rendir, [...]
201	Camara Sony	La camara en si esta genial,la calidad de las fotos es muy buena,incluso con poca luz,zoom muy potente, relacion calidad precio muy buena,peso justo,la unica pega es el tamano,es un poco grande en comparacion con su hermana pequena solamente por el visor.Pero por lo demas perfecta.
301	Cargador AUKEY	Buscaba un cargador que no se calentara, silencioso y que se adaptara a las condiciones de carga rapida de mi Xiaomi Redmi Note 3 Pro. Cumple con todas las exigencias que yo tenia. Lo volveria a comprar.
401	Apple iPhone	Estoy mas que satisfecho con el producto, a pesar de que tenia desconfianza con el, se me presento un problema en el equipo referente al sistema operativo (no al equipo propiamente) y al checar con apple, me di cuenta que tenia la garantia extendida.
501	Kindle Paperwhite	La version Paperwhite, como e-reader, cumple lo que promete, es poco lo que promete, pero lo cumple al 100 %. Llego muy rapido, en buenas condiciones. El producto se siente de calidad y resistencia (no es premium, pero no necesariamente es malo eso) puedes transportarlo facilmente por lo que [...]
601	Tarjeta SanDisk	La tarjeta a respondido muy bien: Las velocidades de lectura son muy altas tal como anuncia el fabricante. Por otro lado las velocidades de escritura son buenas para lo que cuesta esta tarjeta aunque siempre dependera del equipo, numero de archivos, tamano, etc.RESUMIENDO: Muy buena MicroSd.[...]
701	Smartwatch U8	No esperaba mucho cuando ordene este reloj dado su bajo precio y viendo que es un reloj generico que se vende bajo varias marcas pero de todos modos lo compre, no esperaba una fraccion de lo que es el apple watch o un pebble pero si debo admitir que cuando lo recibi y si prendio me alegre jaja.De entrada les [...]
801	Television Samsung	Adquiri este equipo para regalarlo, estoy muy contento con mi compra, la calidad de la imagen es excelente y viene lista para conectarla a internet, la instalacion se realiza muy facil y rapido (5 minutos maximo), debes tener un desarmador a la mano ya que la base viene desarmada y tienes que [...]
901	Halo 5	no es el juego de halo con mejor historia, es superado por todos los anteriores en ese aspecto.pero el multijugador en mi opinion, es el mejor de todos, junto con el de halo 3, e incluso halo 5 es mas equilibrado en ese aspecto

Tabla 6.1: Se muestran algunos de los comentarios capturados de la base de datos de Amazon México para algunos productos. Se han descartado los signos de puntuación como parte del preprocesamiento.

Evidentemente la semántica en esta base de datos es muy variada. Algunos comentarios pueden referirse a características de teléfonos móviles, cámaras, televisiones, etc. Aunque las diversas semánticas en la base ya se conocen a priori (dado que se sabe a qué producto corresponde cada comentario) sería interesante evaluar la capacidad de una estrategia de modelación de tópicos latentes para descubrir por sí misma dichas semánticas. Es decir, para este experimento supondremos que partimos de una base de 1000 opiniones que los clientes han realizado sobre una variedad de productos que ofrece una compañía y se buscará implementar una técnica de modelación de tópicos latentes con el fin de descubrir un patrón semántico en los datos. Este patrón podría ser útil para definir un conjunto de conglomerados que particionen a los clientes en diferentes grupos según su tendencia más fuerte hacia cierto tópico en el espacio simplejo. Estos grupos podrían ser usados después por el área de ventas para implementar campañas estratégicas de marketing.

Nótese que en este caso no se partirá de una base de datos previamente clasificada, sino que por el contrario se buscará que el algoritmo proponga una manera de agrupar los datos a partir de algún patrón que éste identifique en ella. Este enfoque es comúnmente conocido como aprendizaje no supervisado y una técnica clásica dentro de éste es el análisis de conglomerados. A pesar de que existen diversas técnicas de análisis de conglomerados que podrían ser útiles, en este caso se usarán las técnicas de modelación de tópicos latentes presentadas en el capítulo 2. Como se mencionó en el capítulo 1, un modelo de tópicos latentes puede conducir a resultados más interesantes que un análisis de conglomerados cuando se sabe a priori que existe una variedad de tópicos en la base de datos. Para este caso la modelación de tópicos latentes es ideal porque se sabe que existe una variedad de productos sobre los cuales los clientes han comentado.

6.2. Preprocesamiento de la base de datos

La base de datos de Amazon fue capturada como texto plano y por ello será necesario aplicar un tratamiento previo para que los algoritmos de modelación de tópicos latentes puedan operar en ella. En general, una metodología similar a la ejemplificada en la sección 5.2 será suficiente para este caso también. La tabla 6.2 resume las técnicas de preprocesamiento aplicadas a la base de datos de Amazon.

Método	¿Se aplicó?
Remover números	SI
Remover puntuación	SI
Remover términos escasos	SI
n-gramas	NO
Lematización	NO
Remover espacios en blanco	SI
Cambiar a minúsculas	SI
Long. mín por término	SI
Long. máx por término	NO

Tabla 6.2: Resúmen del preprocesamiento aplicado a la base de datos de Amazon

Dado que en este caso los números no aportan información semántica importante se han descartado. La puntuación y los espacios en blanco también se han removido con el fin de evitar inconsistencias por faltas de ortografía en la base de datos. Los términos más escasos también han sido omitidos ya que su ausencia (o presencia) sólo afectará a una pequeña porción de los documentos en el corpus. Aunque algunos trabajos muestran que es posible obtener mejoras modestas en el desempeño del algoritmo al considerar n-gramas, por simplicidad esta técnica no ha sido aplicada en este caso. Por lo general es deseable aplicar una técnica de lematización

como tratamiento previo a la base de datos, sin embargo se ha documentado que esto puede empeorar la calidad del clasificador en algunos casos y no siempre es posible contar con una herramienta que realice esta tarea adecuadamente. En esta aplicación se ha omitido el uso de un lematizador. Las mayúsculas han sido cambiadas por minúsculas con el fin de estandarizar la base de datos y se ha fijado una longitud mínima de dos caracteres por término con el fin de descartar palabras vacías.

Una vez que se han estandarizado los documentos de acuerdo a los procedimientos anteriores, se procede a adoptar la representación matricial de documento-término. La figura 6.3 muestra la matriz de documento-término de la base de datos de Amazon. El resultado final fue una matriz con un léxico de $2,061$ palabras.

Matriz Documento-Término *Amazon*

abajo	abrir	abrirlo	absolutamente	absoluto	aca	acabado	acabados	acabo	...	amantes	amazon	ambientes	ambos	amigable	amigo	amigos	amortiguación	ampliación
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0

Tabla 6.3: Se muestran los primeros seis renglones de la matriz *documento-término* de la base de datos de *Amazon* después de haber removido las palabras vacías y aplicado técnicas de preprocesamiento de texto. También se removieron los términos más escasos, la puntuación, los números y se cambiaron a minúsculas todos los caracteres.

6.3. Experimentos

Con el fin de determinar un modelo de tópicos latentes adecuado, se han llevado a cabo dos experimentos. El primero de ellos tiene como objetivo comparar el rendimiento entre el modelo de asignación Dirichlet latente y el modelo tópico correlacionado. Una vez que se ha determinado cuál de estos dos modelos usar, el segundo de estos experimentos responde a la cuestión del número adecuado de tópicos.

6.3.1. ¿Asignación Dirichlet latente o modelo tópico correlacionado?

A pesar de que el modelo tópico correlacionado constituye una propuesta más interesante, la decisión de cuál modelo usar deberá estar basada siempre en los datos. A pesar de que para algunos casos la estructura de correlación introducida por el modelo tópico correlacionado pueda dar mejores resultados esto no tiene que ser siempre así. Con el fin de determinar qué modelo usar en este caso, se ha entrenado una serie de modelos tópicos. Intuitivamente se esperaría que el modelo tópico óptimo usara 10 tópicos latentes, ya que este es el número de productos que se han considerado para construir la base de datos. En este caso ambos modelos se han entrenado tomando los primeros 20 tópicos latentes y se ha medido la preplejidad predictiva (véase sección

2.6.5.6) con el fin de compararlos. La figura 6.1 muestra los resultados obtenidos.

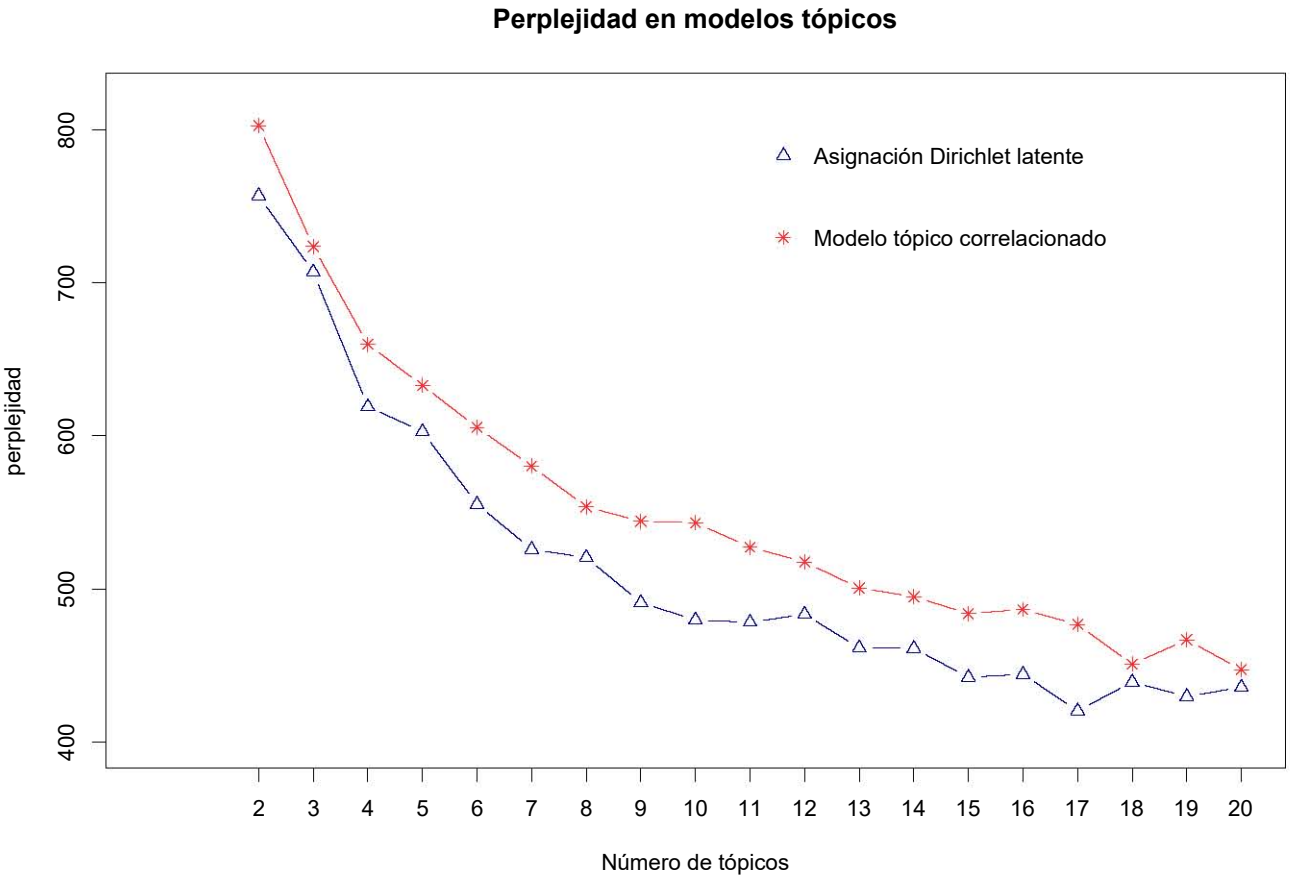


Figura 6.1: Se muestra la perplejidad del modelo de asignación Dirichlet latente y el modelo tópico correlacionado para los primeros 20 tópicos latentes.

Al igual que en muchas aplicaciones, un modelo más sencillo resultó ser en este caso la mejor opción. La figura 6.1 muestra que los mejores resultados son alcanzados por el modelo de asignación Dirichlet latente al reportar los valores más pequeños en la perplejidad. Esto sugiere que consideraciones adicionales en la estructura de correlación entre los tópicos de la base de datos no conducen a un mejor modelo. Esto puede deberse en parte a que en este caso la distribución normal logística no es lo suficientemente flexible para modelar las características particulares en la estructura de correlación entre los tópicos de los datos. Es interesante notar que después de 10 tópicos la perplejidad decrece más lentamente. Esto parece indicar que de 10 a 20 tópicos puede ser la elección más sensata. Dados los resultados obtenidos en esta sección, en lo que resta del capítulo se considerará sólo el modelo de asignación Dirichlet latente.

6.3.2. ¿Cuántos tópicos latentes?

La sección anterior nos permitió determinar que el modelo de asignación Dirichlet latente es el más apropiado para este caso. Sin embargo aún queda pendiente encontrar el número adecuado de tópicos latentes que considerará el modelo. En este experimento se usarán los conceptos introducidos en la sección 2.6.5 con el fin de determinar el número adecuado de tópicos. Para ello se considerarán cuatro medidas: Griffiths (2004), Cao Juan (2009), Arun (2010), Deveaud (2014). La figura 6.2 muestra los resultados obtenidos.

En este caso se puede notar que las métricas de Cao Juan y Griffiths coinciden en que el mejor modelo se obtiene con 13 tópicos latentes. Aunque será necesario probar con algunos otros valores cercanos en este rango, este resultado nos motiva a pensar que el modelo ha detectado al menos los 10 tópicos latentes deseados. Las métricas de Arun y Deveaud resultan no informativas en esta situación. La siguiente sección tiene como objetivo presentar de manera explícita el modelo final y los resultados obtenidos a partir de estos experimentos.

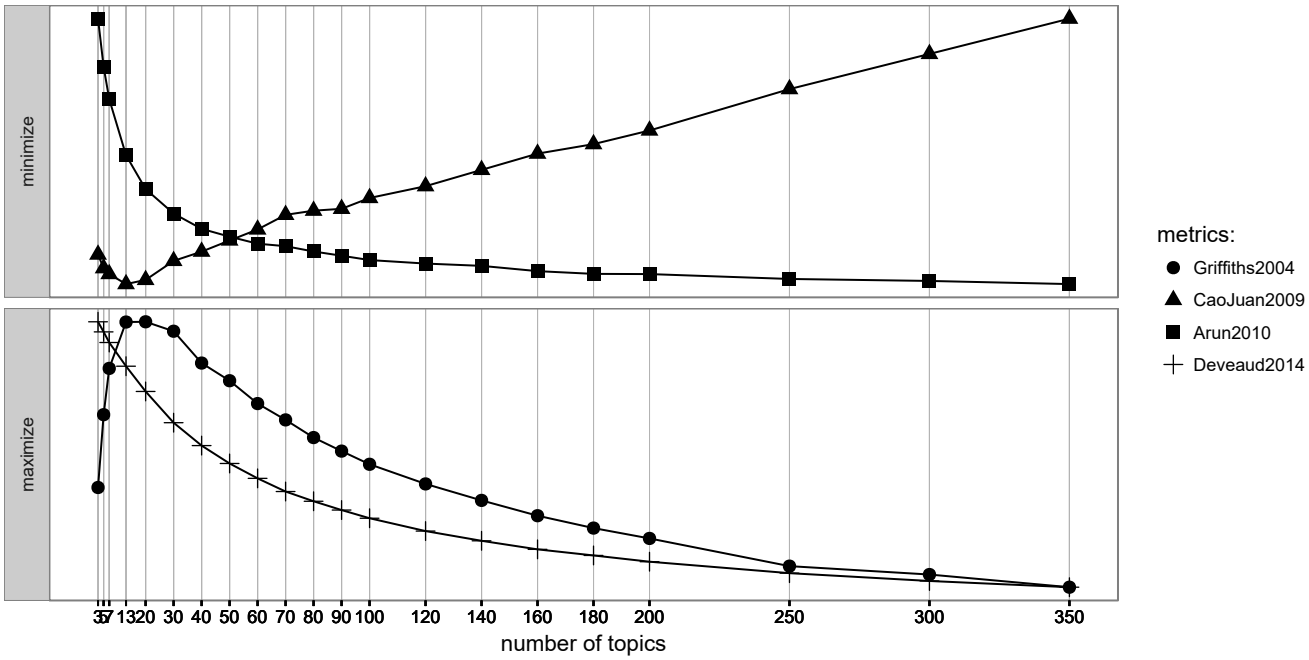


Figura 6.2: Se muestra la implementación de las técnicas de selección de tópicos en la muestra de opiniones de Amazon.

6.4. Visualización de resultados

A partir de la sección anterior fue posible concluir que un modelo de asignación Dirichlet latente con 13 tópicos puede ser el más adecuado. Sin embargo, el proceso de selección y ajuste del modelo es más laborioso que eso e incluye una revisión cuidadosa de los resultados. En general un buen modelo de tópicos latentes es aquel que produce como resultado una variedad de conjuntos semánticos homogéneos en su interior y heterogéneos entre sí. Intuitivamente una lectura rápida de los tópicos debe ser suficiente para identificar la semántica a la que pertenecen. En un modelo tópico mal estructurado la semántica de cada tópico será confusa y difícil de determinar.

Dadas las consideraciones anteriores se exploró una serie de modelos de asignación Dirichlet latente para determinar el más adecuado. Para ello se consideraron de 9 a 20 tópicos latentes y se observó la semántica de los tópicos formados en cada caso. El resultado más interesante se obtuvo con 10 tópicos latentes. Esto, además de coincidir con nuestra intuición, prueba la efectividad de una estrategia de modelación de tópicos latentes para detectar patrones en los datos. Curiosamente cada tópico coincide con la semántica de cada uno de los productos considerados. Los tópicos y sus primeras 30 palabras más importantes se muestran en las tablas 6.4 y 6.5. Los nombres sugeridos para cada uno de los tópicos se muestran entre paréntesis.

Tópicos latentes en base de datos de Amazon				
Tópico 1 (Memoria)	Tópico 2 (Cargador)	Tópico 3 (Batería)	Tópico 4 (Audífonos)	Tópico 5 (Reloj)
tarjeta	carga	bateria	audifonos	reloj
mas	cargador	carga	sonido	puede
marca	cable	mas	calidad	solo
velocidad	rapida	precio	mas	aplicacion
movil	rapido	telefono	precio	android
memoria	mas	celular	excelente	puedo
precio	bateria	producto	cable	bien
adaptador	cargar	bien	bien	producto
capacidad	dos	cargas	bajos	precio
rapida	grande	cargar	audio	funciones
bien	movil	veces	mejor	unico
lectura	usb	puede	estan	notificaciones
compre	bien	iphone	aunque	cuenta
ahora	producto	horas	buenos	mas
problema	bastante	pila	buena	dar
samsung	funciona	dura	agudos	apple
funciona	veces	aunque	buen	caso
datos	tiempo	buena	diseno	hacer
tarjetas	telefono	ser	bastante	iphone
ningun	pesada	ademas	relacion	dura
escritura	vez	cumple	graves	bluetooth
fotos	uso	buen	mejores	llamadas
clase	hace	demasiado	compra	bueno
sandisk	bueno	tiempo	bueno	celular
buena	unico	mejor	marca	funciona
problemas	tan	samsung	volumen	etc
uso	capacidad	forma	menos	cel
almacenamiento	viene	capacidad	sennheiser	mensajes
videos	meses	completamente	suenan	recomiendo
perfectamente	excelente	menos	musica	llega

Tabla 6.4: Se muestran las primeras 30 palabras de los tópicos (1-5) obtenidos bajo un modelo de asignación Dirichlet latente

Tópicos latentes en base de datos de Amazon				
Tópico 6 (Cámara)	Tópico 7 (Videojuego)	Tópico 8 (Televisión)	Tópico 9 (iPhone)	Tópico 10 (Kindle)
fotos	juego	producto	llego	leer
calidad	compre	excelente	producto	kindle
camara	bien	buen	amazon	libros
buena	mas	buena	envio	mas
precio	dia	llego	dia	pantalla
zoom	amazon	precio	tiempo	luz
relacion	pena	calidad	compra	lectura
imagen	llego	compra	excelente	excelente
pilas	cuanto	condiciones	mas	tamano
hace	encanto	recomiendo	bien	dispositivo
buenas	bueno	amazon	comprar	bateria
fotografia	halo	servicio	solo	mejor
mas	buena	bien	nuevo	diferencia
bastante	gran	tiempo	chip	solo
bien	barato	rapido	telcel	puedes
gran	historia	smart	dias	libro
aunque	compro	equipo	asi	perfecto
facil	salio	ampliamente	sellado	producto
recomiendo	medio	imagen	servicio	cualquier
falta	creo	mejor	caja	verdad
unas	vez	satisfecho	venia	noche
hacer	producto	pantalla	precio	vista
compra	buen	facil	entrega	pdf
manual	mejor	entrega	iphone	version
video	envio	ademas	paquete	dura
mala	despues	recomendable	vendedor	puedo
mejor	servicio	funciona	perfecto	comodo
bateria	fecha	dia	telefono	ademas
quieres	verdad	super	problema	practico
contento	rapido	fecha	condiciones	experiencia

Tabla 6.5: Se muestran las primeras 30 palabras de los tópicos (6-10) obtenidos bajo un modelo de asignación Dirichlet latente

Una manera rápida e interesante de visualizar los resultados en los comentarios es identificar cada palabra usada en el comentario con el tópico al que pertenece. Por ejemplo, un comentario que habla sobre audífonos mencionará de manera natural palabras como “audio”, “sonido”, “graves”, “agudos”, etc. y se esperaría que todas estas palabras pertenezcan al tópico 4 (Audífonos). Las siguientes diez figuras muestran algunos de los comentarios de la base de datos de Amazon y sus respectivas palabras etiquetadas con el número del tópico correspondiente. También se han agregado otros metadatos como el autor, la fecha de publicación, el número de estrellas que el usuario otorga al producto, el título del comentario, el nombre del producto y el tópico que prevalece en el comentario. La intensidad del color en las palabras representa su importancia en el comentario.

★★★★★ “Excelente compra”

Por *“Cliente de Amazon”* el 31 de marzo de 2017

Producto: Sennheiser CX 300 II Precision Audifonos de boton con realce de graves, color negro

Excelente[4] calidad[4] de audio[4], presencia[4] en los bajos[4] sin opacar medios[4] y agudos[4]. Buen[4] aislamiento[4] del ruido[4] externo. Personalmente[4] no me agrado[4] el diseno[4], el cable[4] del auricular[4] derecho[4] es mas[4] largo[4] para que lo pases detras[4] del cuello[4]. !Amazon[4] cumplio[4] anticipadamente con la entrega[4]!

Tópico que prevalece: 4 (Audífonos)

★★★★★ “Excelente banco de energia”

Por *“Cliente de Amazon”* el 20 de diciembre de 2016

Producto: ADATA APT100 - Bateria de 10000 mAh, negro y verde

La verdad[2] me sorprendio[3] mucho el tamano[2] de la bateria[2] pero despues[2] de probar[2] su desempeno[2], creo[2] que es lo menos[3] importante[3]. Tengo un Samsung[3] Galaxy[3] y pude[2] recargarlo 5 veces[2] y media[3]. Al momento[2] de la entrega[9], ya viene[2] con una carga[2] completa[2] la bateria[2]. Una vez[2] vacia, tardo[9] entre 3 y 5 horas[3] para cargarse[2] completamente.Posiblemente solo[2] deban agregar[3] al producto[2] algun[2] cable[2] para conectar[2] los aparatos[2] que vayas a recargar[2]. Pero por lo demas[2], el producto[2] es perfecto[2].

Tópicos que prevalecen: 2,3 (Cargador, Batería)

★★★★★ “Una interesante opcion para iniciearse en la fotografia”

Por *“Pedro L.”* el 20 de abril de 2017

Producto: Sony DSC-H300 - Camara compacta de 20.1 MP (pantalla de 3, zoom optico 35x, estabilizador de imagen electronico, video HD 720p), negro

Buscaba[6] una camara[6] para iniciar[6] a un familiar[6] en la fotografia[6], esta camara[6] cumple[6] perfectamente[6] su cometido[6]. sencilla[6] de manejo[6] y con unas[6] funciones[6] lo suficientemente[6] amplias y a la vez[6] sencillas como para cumplir[6] su cometido.La calidad[6] de las imagenes[6] es muy buena[6], su flash[6] ilumina bastante[6] bien[6]. El zoom[6] es perfecto[6] para empezar.La unica[6] pega[6] que podria[6] ponerle es en el disparo a rafagas, algo lento[6]. Pero es suficiente[6] para empezar.Muy recomendable[6].

Topico que prevalece: 6 (Cámara)

★★★★☆ "Buen cargador rapido"

Por *"Cesar"* el 29 de marzo de 2017

Producto: AUKEY Quick Charge 3.0 Cargador de Red 18W [Qualcomm Certificado] para iPhone, Nexus, ect. (Negro)

Realmente[2] se nota[2] la velocidad[2] de carga[2], siempre[2] que tu movil[2] sea compatible[2] con elloTengo un ASUS Zenfone 2 y pese[2] a soportar[2] la carga[2] rapida[2], no incluye[2] un cargador[2] asi[2] en la caja[2] (fallo[2] imperdonable de ASUS)Por eso compre[2] este cargador[2] y la verdad[2] que estoy muy satisfecho[2]

Topico que prevalece: 2 (Cargador)

★★★★★ "Excelente servicio de Amazon"

Por *"Manuel F"* el 14 de agosto de 2015

Producto: Apple iPhone 6 16 GB, Oro Desbloqueado

Pedi[9] la entrega[9] de un dia[9]. Al final[9] obtuve el producto[9] en menos[3] de 24 horas[3] desde que realice[3] el pedido.En cuanto[9] al producto[9], trae[3] una tarjeta[9] SIM[9] de Telcel[9] incluida[9], no estorba pero no la voy[3] a utilizar[5]. El iPhone[9] viene[9] desbloqueado[9] asi[9] que es compatible[5] con cualquier[9] compania[9]. El precio[9] es excelente[9], no lo encontraran[5] a un precio[9] mas[9] bajo[5] en ninguna[3] otra parte[9]. En Amazon[9] puedes[5] comprar[9] el de 64 GB por el mismo[3] precio[9] que conseguirias el de 16 GB en cualquier[9] otro lado[3]. Y el de 16 GB, 2,500 pesos[3] mas[9] barato[9] que en cualquier[9] otro lado[3]. Definitivamente[9] les recomiendo[3] adquirirlo por AMAZON[9] MX.

Topico que prevalece: 9 (Teléfono)

★★★★★ "El mejor lector"

Por *"Cliente de Amazon"* el 21 de marzo de 2017

Producto: Kindle Paperwhite, pantalla E-ink de alta resolucion, luz integrada, color Negro, Wi-Fi

Simplemente[10] el mejor[10] lector[10] del mercado[10]. Tiene muy buenas[10] funciones[5], ha de cuenta[5] que lees[10] en un libro[10] pero sin olor[10], y se puede[10] usar[10] hasta una mano[10] es bastante[10] ligero[10]. muy buena[6] construccion[5]. La funcion[10] de diccionario[10] es de las mejores[10], sirve[5] pues[10] entender[10] mas[10] de lo que lees[10] y de facil[10] acceso[10] cosa[10] que no brindan lo libros[10] de papel[10]. Comienzas tus lecturas[10] donde te quedaste, puedes[10] ampliar[10] el tamano[10] de letra[10] de acuerdo[10] a tus necesidades[10], la retroilumacion es algo muy bueno[10] para ambientes[10] con poca[10] luz[10], y la autonomia[10] cumple[10] y es muy duradera. Lo recomiendo[10] para las personas[10] que NA sean amantes[10] de leer[10], y que pues[10] no sientan diferencia[10] entre un libro[10] y este aparato[5], a la excepticas no. Cuenta[5] tambien[10] con la tienda[10] y muchas[10] paginas[10] para descargar[5] libros[10] en el formato[10] propio[10] del kindle[10].

Topico que prevalece: 10 (Kindle)

★★☆☆☆ "DEJO DE FUNCIONAR A LOS 11 MESES"

Por *"grandu"* el 20 de abril de 2017

Producto: SanDisk Ultra - Tarjeta de memoria Android 32 GB microSDHC UHS-I con adaptador SD, velocidad de lectura hasta 80 MB/s y Clase 10, FFP

Compre[1] esta tarjeta[1] hace[1] 11 meses[1] para ampliar[1] la memoria[1] de mi Moto[1] X Style, NA funciono[2] bastante[1] bien[1], escritura[1] y lectura[1] bastante[1] rapidas[1] y fluidas hasta que hace[1] una semana[1] de repente el movil[1] dejo[2] de reconocer[1] la tarjeta[1], habia[1] dejado[2] de funcionar[2], NA la probe[7] en un par[1] de dispostivos mas[1] por si el error[1] era del movil[1] y tampoco[1] funcionaba[1]. Afortunadamente[7] google[1] fotos[1] me ha salvado al tener[1] todo en la nube.Un 10 a Amazon[7] ya que me la ha reemplazado, pero ya no se si fiarme mucho, he perdido[1] muchos archivos[1].

Topico que prevalece: 1 (Memoria)

★★☆☆☆ "Muy mal producto"

Por *"Cesar Gomez"* el 21 de enero de 2017

Producto: Smartwatch U8 Bluetooth iphone Android Reloj Inteligente

La verdad[5] me parecio[5] muy mal[5] producto[5], la pila[5] le dura[5] muy poco, el bluetooth[5] se desconecta[5] a cada[5] rato[5], las funciones[5] son muy pocas[5], solo[5] llamar y escuchar[5] musica[5] son utiles[5], las bocinas[5] son muy malas[5], no se pueden[5] visualizar[5] las notificaciones[5] aun[5] con la app[5] instalada[5] en el Android[5], cuando te llega[5] alguna[5] el reloj[5] solo[5] vibra[5], no tienes acceso[5] a redes[5] sociales y la interfaz[5] del reloj[5] es muy poco atractiva, etc[5], etc[5]. En fin[5], sinceramente[5] no lo recomiendo[5]. Eso si, el peoducto me llego[5] antes de lo esperado[5].

Topico que prevalece: 5 (Reloj)

★★☆☆☆ "Audio no muy bueno"

Por *"Cliente de Amazon"* el 30 de marzo de 2017

Producto: Samsung UN-32J4300AF- Television LED 32¨(Smart TV)

Pantalla[8] a buen[8] precio[8] pero el audio[4] no es muy bueno[8], lo bueno[8] es que es smart[8] y tiene WiFi[8]

Topico que prevalece: 8 (Televisión)

★★☆☆☆ "La mejor saga de Xbox; no puede faltar."

Por *"Jonathan Pena Pizana"* el 8 de enero de 2016

Producto: Halo 5: Guardians Estandar en Espanol - Xbox One - Standard Edition

El juego[7] esta increible[7], los graficos[7] excepcionales y la historia[7] continua tan[7] buena[7] como en la cuarta entrega!Lo unico[7] que no me agrado[7] es que ya no tiene multijuagador local; me encantaba aventarme partidas[7] por horas[7] con mis hermanos y ahora[7] no ha salido[7] uan justificacion para tener[7] mas[7] de un control[7] en la consola[7]. Aun[7] asi[7] lo recomiendo[7]. **Topico que prevalece: 7 (Videojuego)**

6.5. Conclusiones

A través de esta aplicación se ejemplificó la efectividad y flexibilidad de un modelo de tópicos latentes en un contexto que puede ser de interés para organizaciones dedicadas al comercio de bienes y servicios. Es importante evidenciar la eficacia del enfoque Bayesiano para atacar este tipo de problemas aún en contextos

de alta dimensionalidad. Aunque en este caso se consideró una base de datos conformada por opiniones sobre diversos productos ofrecidos en la plataforma de Amazon, el experimento puede ser replicado en otros escenarios que incluyen redes sociales, foros de discusión o en general cualquier medio digital en el que los usuarios depositen sus comentarios.

Como se mencionó al inicio de esta aplicación, los resultados obtenidos pueden ser utilizados con diferentes fines. Un uso particular podría ser implementar una campaña de marketing de manera más efectiva o simplemente obtener una dimensión reducida de los datos con el fin de entrenar diversos algoritmos de aprendizaje en ella. Un aspecto interesante que puede ser aprovechado es que los resultados obtenidos pueden usarse para definir conglomerados de manera "suave" y ofrecer productos relacionados a los que el cliente ya ha comprado. Por ejemplo, el comentario "Excelente banco de energia" fue hecho por un cliente que compró una batería (tópico 3), sin embargo el modelo también detectó automáticamente que el comentario está relacionado en el tópico de los cargadores (tópico 2). De esta manera la compañía podría mandarle a este cliente información publicitaria tanto de baterías como de cargadores y conseguir un mayor impacto.

Conclusiones

A través de este trabajo se exploraron algunas de las técnicas básicas de minería de textos. La teoría y las aplicaciones propuestas pueden servir para obtener un panorama introductorio en esta área. Las exposiciones de los temas y una gran variedad de aplicaciones pueden ser encontradas en los diversos libros y artículos de investigación disponibles, muchos de los cuales ya han sido incluidos dentro de la bibliografía.

Otro aspecto que se procuró tener en cuenta y ejemplificar a lo largo de las aplicaciones propuestas es la interdisciplinariedad de la minería de textos con otras áreas como las ciencias de la computación y la estadística. Ya que los problemas de minería de textos pueden darse en muy diversos contextos (tales como las ciencias sociales o las ciencias biológicas), es importante contar con un experto que pueda aportar información adicional sobre el problema.

Una de las motivaciones principales que dieron origen a este trabajo fue el deseo por explorar otras formas de datos que han surgido de manera natural junto con los recientes avances tecnológicos. Se espera que este trabajo pueda promover la investigación y el desarrollo de la estadística en esta área con el fin de dar solución a los nuevos retos que tendrán que enfrentar las próximas generaciones de científicos.

A pesar de que el problema de la detección de SPAM ya ha sido tratado en una gran cantidad de libros y artículos, se espera que el lector se haya sentido motivado al rescatar este tema en el ámbito de la telefonía móvil. Estas herramientas que hoy en día conocemos como teléfonos inteligentes han logrado penetrar en nuestras vidas al grado de hacernos vulnerables a un sinfín de amenazas y es necesario que a la par se desarrollen nuevas tecnologías que nos permitan proteger y resguardar nuestra información.

Evidentemente el avance tecnológico no sólo ha traído incertidumbre e inseguridad a nuestras vidas. Es posible incluso atreverse a afirmar que serán más los beneficios y las ventajas a futuro que los problemas y los riesgos, siempre y cuando las nuevas tecnologías se usen apropiadamente. Después de todo no hay que olvidar que el mismo desarrollo tecnológico fue el responsable de la creación de una bomba atómica en algún momento. Sin embargo, cosas increíbles son posibles ahora gracias a la tecnología.

Hasta hace algunos años el desarrollo práctico de la estadística Bayesiana estaba limitado a un modesto número de modelos que servían para abordar los procesos de inferencia desde otro punto de vista. Hoy en día la estadística Bayesiana se ha consolidado como una de las áreas más prometedoras dentro de la estadística y una gran cantidad de modelos se han aplicado a problemas reales gracias al desarrollo de la tecnología. Un ejemplo de la flexibilidad de la modelación bajo el enfoque Bayesiano en diferentes contextos fue ejemplificado en este trabajo a través de la aplicación de los modelos de tópicos latentes.

Es difícil imaginar qué otras cosas pueden ser más difíciles de modelar para un estadístico además de los sentimientos y los pensamientos humanos. Las aplicaciones aquí mostradas ejemplifican sólo algunas de las dificultades de esta tarea. A pesar de que el cerebro ha sido uno de los órganos más estudiados y fascinantes no ha terminado de comprenderse. En el mejor de los casos nos tendremos que defender ante las críticas a nuestros intentos por modelar su comportamiento con la famosa frase “Todos los modelos son erróneos, pero algunos son útiles” que en algún momento mencionó el proclamado estadístico George Box.

Bibliografía

[1] "IBM ¿qué es big data?." https://www.ibm.com/developerworks/ssa/local/im/que-es-big-data/index.html. 2017-08-01.

[2] C. C. Aggarwal and C. Zhai, *Mining text data*. Springer Science & Business Media, 2012.

[3] C. C. Aggarwal, *Data mining: the textbook*. Springer, 2015.

[4] B. Liu, "Sentiment analysis and opinion mining," *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[5] S. M. Weiss, N. Indurkhya, and T. Zhang, *Fundamentals of predictive text mining*, vol. 41. Springer, 2010.

[6] S. Sirmakessis, *Text mining and its applications: results of the NEMIS Launch Conference*, vol. 138. Springer, 2012.

[7] S. Sumathi and S. Sivanandam, "Introduction to data mining principles," *Introduction to Data Mining and its Applications*, pp. 1–20, 2006.

[8] F. Coenen, "Data mining: past, present and future," *The Knowledge Engineering Review*, vol. 26, no. 1, pp. 25–29, 2011.

[9] F. Yongjian, "Data mining: tasks, techniques and applications," *IEEE Potentials*, vol. 16, no. 4, pp. 18–20, 1997.

[10] M. Grobelnik, D. Mladenic, and N. Milic-Frayling, "Text mining as integration of several related research areas: report on kdd's workshop on text mining 2000," *ACM SIGKDD Explorations Newsletter*, vol. 2, no. 2, pp. 99–102, 2000.

[11] I. H. Witten, "Text mining.," 2004. Computer Science, University of Waikato, Hamilton, New Zealand.

[12] G. Miner, *Practical text mining and statistical analysis for non-structured text data applications*. Academic Press, 2012.

[13] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[14] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[15] E. Basegmez, "The next generation neural networks: Deep learning and spiking neural networks," in *Advanced Seminar in Technical University of Munich*, pp. 1–40, 2014.

[16] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," *Shape, contour and grouping in computer vision*, pp. 823–823, 1999.

[17] R. Johnson and T. Zhang, "Effective use of word order for text categorization with convolutional neural networks," *arXiv preprint arXiv:1412.1058*, 2014.

[18] D. Britz, "Implementing a cnn for text classification in tensorflow," 2015.

[19] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification.," in *AAAI*, vol. 333, pp. 2267–2273, 2015.

[20] C. N. Dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts.," in *COLING*, pp. 69–78, 2014.

[21] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, "Text classification using string kernels," *Journal of Machine Learning Research*, vol. 2, no. Feb, pp. 419–444, 2002.

[22] A. Karatzoglou and I. Feinerer, "Text clustering with string kernels in r," in *Advances in Data Analysis*, pp. 91–98, Springer, 2007.

[23] M. W. Berry and J. Kogan, *Text mining: applications and theory*. John Wiley & Sons, 2010.

[24] M. Hofmann and A. Chisholm, *Text Mining and Visualization: Case Studies Using Open-Source Tools*. Chapman and Hall/CRC, 2015.

[25] A. N. Srivastava and M. Sahami, *Text mining: Classification, clustering, and applications*. CRC Press, 2009.

[26] C. L. Blake and C. J. Merz, "UCI repository of machine learning databases [http://www. ics. uci. edu/˜ mlearn/mlrepository. html]. irvine, ca: University of California," *Department of Information and Computer Science*, vol. 55, 1998.

[27] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (methodological)*, pp. 1–38, 1977.

[28] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[29] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian data analysis*, vol. 2. CRC press Boca Raton, FL, 2014.

[30] J. M. Bernardo, "Bruno de Finetti en la estadística contemporánea," *Historia de la Matemática en el siglo XX, S. Rios (ed.), Real Academia de Ciencias, Madrid*, pp. 63–80, 1998.

[31] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proceedings of the National Academy of Sciences*, vol. 101, no. suppl 1, pp. 5228–5235, 2004.

[32] D. M. Blei and J. D. Lafferty, "A correlated topic model of science," *The Annals of Applied Statistics*, pp. 17–35, 2007.

[33] S. Chib and E. Greenberg, "Understanding the Metropolis-Hastings algorithm," *The American Statistician*, vol. 49, no. 4, pp. 327–335, 1995.

[34] D. M. Blei and J. D. Lafferty, "Dynamic topic models," in *Proceedings of the 23rd International Conference on Machine learning*, pp. 113–120, ACM, 2006.

[35] R. E. Kass and A. E. Raftery, "Bayes factors," *Journal of the American Statistical Association*, vol. 90, no. 430, pp. 773–795, 1995.

[36] J. Cao, T. Xia, J. Li, Y. Zhang, and S. Tang, "A density-based method for adaptive lda model selection," *Neurocomputing*, vol. 72, no. 7, pp. 1775–1781, 2009.

[37] R. Arun, V. Suresh, C. Veni Madhavan, and M. Narasimha Murthy, "On finding the natural number of topics with latent Dirichlet allocation: Some observations," *Advances in Knowledge Discovery and Data Mining*, pp. 391–402, 2010.

[38] R. Deveaud, E. SanJuan, and P. Bellot, "Accurate and effective latent concept modeling for ad hoc information retrieval," *Document numérique*, vol. 17, no. 1, pp. 61–84, 2014.

[39] H. M. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno, "Evaluation methods for topic models," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1105–1112, ACM, 2009.

[40] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, p. 391, 1990.

[41] T. Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 50–57, ACM, 1999.

[42] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[43] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2012.

[44] M. Tipping, "Relevance vector machine," Oct. 14 2003. US Patent 6,633,857.

[45] "Relevance Vector Machine (RVR and RVC)." `https://github.com/AmazaspShumik/sklearn-bayes/blob/master/ipython_notebooks_tutorials/rvm_ard/rvm_demo.ipynb`. 24-11-2017.

[46] D. J. MacKay, "Bayesian interpolation," *Neural computation*, vol. 4, no. 3, pp. 415–447, 1992.

[47] M. J. Kearns and U. V. Vazirani, *An introduction to computational learning theory*. MIT press, 1994.

[48] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[49] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[50] M. Minsky and S. Papert, "Perceptrons.," 1969.

[51] D. Williams and G. Hinton, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–538, 1986.

[52] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[53] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[54] M. Kearns, "Thoughts on hypothesis boosting," *Unpublished manuscript*, vol. 45, p. 105, 1988.

[55] L. G. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.

[56] R. E. Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[57] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.

[58] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Icml*, vol. 97, pp. 412–420, 1997.

[59] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, no. Jun, pp. 211–244, 2001.

[60] T. Fletcher, "Relevance vector machines explained," *University College London: London, UK*, 2010.

[61] D. Barber, *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[62] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[63] S. Marsland, *Machine learning: an algorithmic perspective*. CRC press, 2015.

[64] A. Zheng, "Evaluating machine learning models," 2015.

[65] L. Zhang, J. Zhu, and T. Yao, "An evaluation of statistical spam filtering techniques," *ACM Transactions on Asian Language Information Processing (TALIP)*, vol. 3, no. 4, pp. 243–269, 2004.

[66] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[67] "Understanding roc curves." `https://http://www.navan.name/roc/`. 24-11-2017.

[68] Machine Learning Repository, "`archive.ics.uci.edu/ml/datasets/sms+spam+collection`," 28 de Mayo de 2017.

[69] Machine Learning Repository, "`archive.ics.uci.edu/ml/datasets/spambase`," 28 de Mayo de 2017.

[70] M. Hollander, D. A. Wolfe, and E. Chicken, *Nonparametric statistical methods*. John Wiley & Sons, 2013.

[71] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, "Text classification using string kernels," *Journal of Machine Learning Research*, vol. 2, no. Feb, pp. 419–444, 2002.

[72] M. Rafi and M. S. Shaikh, "A comparison of SVM and RVM for document classification," *arXiv preprint arXiv:1301.2785*, 2013.

[73] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe, "Predicting elections with twitter: What 140 characters reveal about political sentiment.," *ICWSM*, vol. 10, no. 1, pp. 178–185, 2010.

[74] A. Tripathy, A. Agrawal, and S. K. Rath, "Classification of sentimental reviews using machine learning techniques," *Procedia Computer Science*, vol. 57, pp. 821–829, 2015.

[75] C. Rain, "Sentiment analysis in Amazon reviews using probabilistic machine learning," *Swarthmore College*, 2013.

[76] G. Casella and E. I. George, "Explaining the gibbs sampler," *The American Statistician*, vol. 46, no. 3, pp. 167–174, 1992.